Augmenting Small-size Tabular Data with Class-Specific Energy-Based Models

Andrei Margeloiu[1]*, **Xiangjian Jiang**[1]*, **Nikola Simidjievski**[2,1], **Mateja Jamnik**[1]
[1]Department of Computer Science and Technology, University of Cambridge, UK
[2]PBCI, Department of Oncology, University of Cambridge, UK
{am2770, xj265, ns779, mj201}@cam.ac.uk

Abstract

Data collection is often difficult in critical fields such as medicine, physics, and chemistry, yielding typically only small tabular datasets. However, classification methods tend to struggle with these small datasets, leading to poor predictive performance. Increasing the training set with additional synthetic data, similar to data augmentation in images, is commonly believed to improve downstream tabular classification performance. However, current tabular generative methods that learn either the joint distribution $p(\mathbf{x}, y)$ or the class-conditional distribution $p(\mathbf{x} \mid y)$ often overfit on small datasets, usually worsening classification performance compared to using real data alone. To solve these challenges, we introduce TabEBM, a novel class-conditional generative method using Energy-Based Models (EBMs). Unlike existing tabular methods that use a shared model to approximate all class-conditional densities, our key innovation is to create distinct EBM generative models for each class, each modelling its class-specific data distribution individually. This approach creates robust energy landscapes, even in ambiguous class distributions. Our experiments show that TabEBM generates synthetic data with higher utility than existing methods. When used for data augmentation, our synthetic data consistently improves the classification performance across diverse datasets of various sizes, especially small ones. Code is available at https://github.com/andreimargeloiu/TabEBM.

1 Introduction

In scientific fields such as medicine, physics, and chemistry, collecting tabular data is often challenging due to the experimental nature of data acquisition [4, 42, 3, 26, 61, 10]. Due to the small size of such datasets [4, 38], training machine learning models that can aid in tasks such as disease diagnosis [44, 31], material property prediction [29], and chemical compound classification [9], often suffer from poor performance [60, 44, 31]. In the case of vision and language tasks, a standard remedy to data scarcity is employing data augmentation techniques [58, 59, 48, 57]. However, applying data augmentation to tabular data remains understudied, as tabular datasets are very diverse and lack explicit symmetries [6], such as rotations or translations seen in images. Consequently, existing tabular data augmentation methods often yield mixed results and can even degrade model performance [43, 57, 40], hindering their widespread adoption.

Tabular augmentation typically involves training generative models to approximate either the joint distribution $p(\mathbf{x}, y)$ [68, 18] or the class-conditional distribution $p(\mathbf{x}|y)$ [68, 34, 66, 39, 40]. A key challenge of joint distribution methods is maintaining the original label distribution, as such generators can fail to generate data for specific classes (see Appendix D for an example). On

*Equal contribution.

Table Representation Learning Workshop at NeurIPS 2024.

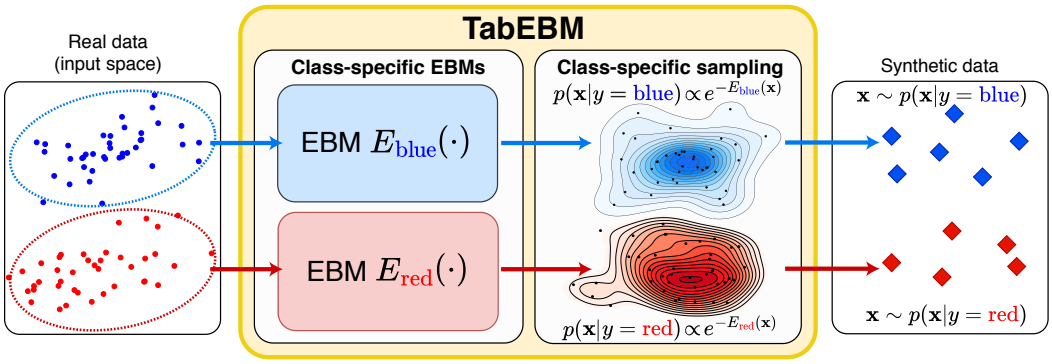


Figure 1: **An overview of TabEBM**. We learn *distinct* class-specific Energy-Based Models (EBMs) $E_{\text{blue}}(\mathbf{x})$ and $E_{\text{red}}(\mathbf{x})$ exclusively on the points of their respective class. Each EBM approximates a class-conditional distribution $p(\mathbf{x}|y)$. TabEBM allows synthetic data generation by sampling from the estimated distributions for each class $p(\mathbf{x}|y = \text{blue})$ and $p(\mathbf{x}|y = \text{red})$.

the other hand, while class-conditional models learning $p(\mathbf{x}|y)$ preserve the stratification of the original data, they often employ a *shared* model to represent all class-conditional densities. This, however, can lead to overfitting, particularly in imbalanced datasets where the model may prioritise more frequent classes [17], ignoring unique features needed for generating label-invariant samples. Additionally, in datasets with limited data, this can lead to mode collapse [55, 56], where the model does not effectively capture the diversity of each class [56], and thus tends to perform poorly in a multi-class setting. We further provide an extended discussion on related work in Appendix A.

To address the challenges of class-conditional tabular generation, we introduce TabEBM (Figure 1), a new method for tabular data augmentation utilising Energy-Based Models (EBMs). Our method introduces two innovations: (i) *Distinct class-specific models:* TabEBM constructs a collection of individual models – one for each class – which, by design, enables learning distinct marginal distributions for the inputs associated with each class. This, in turn, enables performing data augmentation while maintaining the original label distribution. (ii) *Generative models:* we build novel class-specific generators that produce high-quality synthetic data even from extremely few samples. Specifically, we create a surrogate binary classification task for each class and fit it with a pre-trained tabular in-context classifier. We then convert the binary classifier into an EBM, a generative model, without additional training. Using class-specific EBMs makes the energy landscape more robust to class overlaps, compared to using a single shared EBM to approximate the class-conditional distribution.

2 TabEBM

Notation. We address classification problems with C classes, denoted by $\mathcal{Y} = \{1, 2, \ldots, C\}$. Let $\{(\mathbf{x}^{(i)}, y_i)\}_{i=1}^{N}$ represent a dataset of N samples, each being a D-dimensional vector $\mathbf{x}^{(i)} \in \mathbb{R}^D$, with a corresponding label $y_i \in \mathcal{Y}$. For each class $c \in \mathcal{Y}$, we define $\mathcal{X}_c = \{\mathbf{x}^{(i)} \mid y_i = c\}$ as the subset of samples labelled with class c. Let $f_\theta(\cdot)$ denote a classifier. The expression $f_\theta(\mathbf{x})[y]$ represents the (unnormalised) logit assigned to the class y for the input $\mathbf{x}$.

2.1 Preliminaries on Energy-Based Models

An Energy-Based Model (EBM) [35] defines a probability density function $p_\theta(\mathbf{x})$ through an energy function $E(\mathbf{x})$. Specifically, the model posits that $p(\mathbf{x}) \propto e^{-E(\mathbf{x})}$, where $E(\mathbf{x})$ represents the unnormalised negative log-density of the input $\mathbf{x}$. An important observation is that energy-based models can utilise the same model architectures as standard classification models [23]. Typically, the logits $f_\theta(\mathbf{x})[y]$ are reinterpreted to define an energy-based model for the joint distribution $p(\mathbf{x}, y)$. Furthermore, the energy function for the marginal distribution $p(\mathbf{x})$ is obtained by marginalising over $p(\mathbf{x}, y)$, resulting in $E(\mathbf{x}) = -\text{LogSumExp}_{y'} f_\theta(\mathbf{x})[y']$.

2.2 Distinct Class-Specific Energy-Based Models

TabEBM is a class-conditional generative model $p(\mathbf{x}|y)$ implemented using a set of EBMs, $\{E_1(\mathbf{x}), E_2(\mathbf{x}), \ldots, E_C(\mathbf{x})\}$. Our approach assumes that the class-conditional density $p(\mathbf{x}|y = c)$ is best modelled using its class-specific data $\mathcal{X}_c$. Thus, for each class c, we construct a class-specific EBM, $E_c(\mathbf{x})$, using only the data from that class, $\mathcal{X}_c$, such that $p(\mathbf{x}|y = c) \propto \exp(-E_c(\mathbf{x}))$.

Model training ("Class-specific EBMs" in Figure 1). Building TabEBM requires training multiple classifiers on a novel task and reinterpreting their logits. For each class c, we propose a *surrogate binary classification task* to determine if a sample belongs to class c by comparing $\mathcal{X}_c$ against a set of surrogate negative samples $\mathcal{X}_c^{\text{neg}}$. Specifically, we generate the negative samples at the corners of a hypercube in R^D. For each dimension d, the coordinates of a negative sample are either $\alpha_{\text{dist}}^{\text{neg}}\sigma_d$ or $-\alpha_{\text{dist}}^{\text{neg}}\sigma_d$, where $\alpha_{\text{dist}}^{\text{neg}}$ is a fixed constant and σ_d is the standard deviation of dimension d. In Appendix E.1, we provide a 2D example along with several ablations, demonstrating that the placement of negative samples is critical for achieving an accurate energy function.

We create the combined dataset $\mathcal{D}_c$ for the surrogate binary classification task by labelling $\mathcal{X}_c$ as 1 and $\mathcal{X}_c^{\text{neg}}$ as 0: $\mathcal{D}_c = (\mathcal{X}_c \cup \mathcal{X}_c^{\text{neg}}, \{1\}^{|\mathcal{X}_c|} \cup \{0\}^{|\mathcal{X}_c^{\text{neg}}|})$. We then train a binary classifier $f_\theta^c(\cdot)$ on $\mathcal{D}_c$ and use it to construct the class-specific energy $E_c(\mathbf{x})$ for class c. To do this, we reinterpret the logits $\{f_\theta^c(\mathbf{x})[0], f_\theta^c(\mathbf{x})[1]\}$ of the trained binary classifier as components of an approximated joint distribution for the surrogate binary task. Next, we approximate $p_c(\mathbf{x})$ by marginalisation:

$$
\begin{aligned}
p_c(\mathbf{x}) &= p_c(\mathbf{x}, 0) + p_c(\mathbf{x}, 1) \\
&= \frac{\exp\left(\log\left(\exp(f_\theta^c(\mathbf{x})[0]) + \exp(f_\theta^c(\mathbf{x})[1])\right)\right)}{Z} && (Z \text{ is the normalisation constant}) \\
\rightarrow E_c(\mathbf{x}) &= -\log\left(\exp(f_\theta^c(\mathbf{x})[0]) + \exp(f_\theta^c(\mathbf{x})[1])\right) && (\text{TabEBM class-specific energy}) \qquad (1)
\end{aligned}
$$

For the binary classifier $f_\theta^c(\cdot)$ in the surrogate binary classification, we use TabPFN [27], a pre-trained tabular in-context model. In this context, "training" the TabPFN classifier is analogous to the K Nearest Neighbour algorithm, which simply performs inference based on the training data.

Data generation ("Class-specific sampling" in Figure 1). TabEBM generates data in two steps. First, we sample a class c from the empirical distribution $c \sim p(y)$. Then, we sample a data point $\mathbf{x}$ from the conditional distribution $\mathbf{x} \sim p(\mathbf{x}|y = c)$ approximated by the class-specific energy-based model $E_c(\mathbf{x})$, as outlined in Algorithm 1 (see Appendix B). We employ Stochastic Gradient Langevin Dynamics (SGLD) [67] to perform this sampling. SGLD is an efficient method for high-dimensional data, combining stochastic gradient descent with Langevin dynamics. The update rule for SGLD at each iteration is $\mathbf{x}_{t+1} = \mathbf{x}_t - \frac{\eta}{2}\nabla E(\mathbf{x}_t) + \epsilon_t$, where $\epsilon_t \sim \mathcal{N}(0, \eta\mathbf{I})$ is a Gaussian noise that introduces randomness into the sampling process, enhancing the exploration of the distribution. Appendix E.2 further shows that TabEBM is stable to hyperparameters for the sampling process.

3 Experiments: Can TabEBM improve predictors via data augmentation?

Datasets. We utilise eight open-source tabular datasets from OpenML [5] – across five domains: Medicine, Chemistry, Engineering, Language and Economics. We further vary the degrees of data availability (i.e., N_{real}), leading up to 33 test cases. Appendix C.1 provides detailed descriptions.

Benchmark generators. We compare TabEBM against eight existing tabular data generation methods of eight different categories: SMOTE [11], TVAE [68], CTGAN [68], NFLOW [18], TabDDPM [34], ARF [66], GOGGLE [39] and TabPFGen [40]. Furthermore, we also include a "Baseline" model, where only real data is used to train downstream predictors.

Downstream predictors. We select six representative downstream predictors, including three standard baselines: Logistic Regression (LR) [13], KNN [21] and MLP [22]; two tree-based methods: Random Forest (RF) [8] and XGBoost [12]; and a PFN method: TabPFN [27].

Data augmentation setup. For each dataset of N samples, we first split it into stratified train and test sets. Then we subsample the full train set to simulate different levels of data availability and split each subset into stratified training and validation sets with a ratio of 4:1. Given N_{real} real samples, we first train generators on the real training data and then generate N_{syn} synthetic samples. For training the downstream predictors, we expand the real training split by adding the synthetic samples. The optimal N_{syn} remains an open problem for tabular data [43, 57, 25]. To provide a head-to-head comparison of the effect of data augmentation across subsample sizes, we perform data augmentation with a large synthetic set ($N_{\text{syn}} = 500$) across all test cases, and the synthetic data has the same class distribution as the real training data. We provide detailed descriptions of data splitting in Appendix C.2 and preprocessing in Appendix C.3.

Evaluation metric. We evaluate the effect of using synthetic data for data augmentation with the *balanced accuracy* of downstream predictors. Typically, higher accuracy improvements (i.e., $\text{ACC}_{\text{Generator}} - \text{ACC}_{\text{Baseline}} > 0$) demonstrate better utility of synthetic data for data augmentation.

Table 1: **Classification accuracy** (%) aggregated over six downstream predictors, comparing data augmentation on five real-world tabular datasets with varied real data availability (full results of eight datasets are in Appendix E.3). We report the mean ± std balanced accuracy. We **bold** the highest accuracy for each test case. Our method, TabEBM, consistently outperforms training on real data alone, and achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	$28.14_{\pm 6.83}$	N/A	$21.18_{\pm 1.48}$	$22.00_{\pm 3.43}$	$21.30_{\pm 2.84}$	$22.12_{\pm 5.30}$	$24.82_{\pm 2.88}$	$22.40_{\pm 9.28}$	$33.25_{\pm 5.01}$	**33.84**$_{\pm 4.92}$
		50	$50.72_{\pm 10.53}$	$54.52_{\pm 8.59}$	$39.54_{\pm 5.19}$	$36.32_{\pm 7.17}$	$35.37_{\pm 8.00}$	$35.11_{\pm 11.78}$	$41.99_{\pm 5.24}$	$37.53_{\pm 14.72}$	$54.45_{\pm 7.96}$	**55.91**$_{\pm 6.41}$
		100	$67.83_{\pm 11.72}$	$73.25_{\pm 7.48}$	$59.28_{\pm 7.20}$	$57.64_{\pm 9.95}$	$52.57_{\pm 9.55}$	$56.37_{\pm 9.64}$	$57.01_{\pm 8.56}$	$51.69_{\pm 16.68}$	$71.53_{\pm 9.87}$	**73.31**$_{\pm 6.77}$
		200	$81.66_{\pm 10.18}$	$85.65_{\pm 6.24}$	$76.42_{\pm 7.71}$	$74.88_{\pm 8.20}$	$72.10_{\pm 10.04}$	$75.86_{\pm 9.30}$	$74.07_{\pm 8.74}$	$73.57_{\pm 6.74}$	$84.95_{\pm 7.47}$	**86.14**$_{\pm 5.50}$
		500	$93.49_{\pm 5.28}$	$94.73_{\pm 3.67}$	$92.24_{\pm 3.73}$	$91.48_{\pm 4.43}$	$90.44_{\pm 5.54}$	$90.62_{\pm 5.63}$	$91.79_{\pm 4.53}$	$91.31_{\pm 5.20}$	$94.87_{\pm 3.70}$	**95.18**$_{\pm 3.10}$
	fourier	20	$28.30_{\pm 12.09}$	N/A	$21.32_{\pm 4.06}$	$18.19_{\pm 3.90}$	$17.30_{\pm 3.03}$	$15.35_{\pm 3.26}$	$21.75_{\pm 2.76}$	$16.70_{\pm 2.91}$	$36.72_{\pm 7.30}$	**37.13**$_{\pm 6.01}$
		50	$53.69_{\pm 8.04}$	$55.51_{\pm 7.43}$	$37.96_{\pm 4.48}$	$35.09_{\pm 7.46}$	$31.94_{\pm 8.99}$	$35.99_{\pm 13.06}$	$40.32_{\pm 6.70}$	$33.56_{\pm 14.02}$	$55.11_{\pm 10.66}$	**56.57**$_{\pm 7.12}$
		100	$63.70_{\pm 6.76}$	$64.10_{\pm 6.89}$	$50.46_{\pm 8.61}$	$49.26_{\pm 9.15}$	$44.58_{\pm 8.40}$	$52.79_{\pm 10.04}$	$51.13_{\pm 6.35}$	$41.93_{\pm 15.60}$	$63.86_{\pm 7.76}$	**65.21**$_{\pm 6.42}$
		200	$70.99_{\pm 4.88}$	$71.43_{\pm 4.47}$	$62.17_{\pm 7.29}$	$62.92_{\pm 7.87}$	$59.15_{\pm 8.33}$	$68.05_{\pm 6.91}$	$62.53_{\pm 6.97}$	$56.44_{\pm 10.13}$	$71.81_{\pm 5.35}$	**72.36**$_{\pm 3.77}$
		500	$77.72_{\pm 2.36}$	$77.51_{\pm 2.60}$	$73.29_{\pm 4.97}$	$74.61_{\pm 4.89}$	$71.74_{\pm 6.54}$	$77.04_{\pm 3.64}$	$74.31_{\pm 4.40}$	$70.61_{\pm 6.01}$	$77.15_{\pm 2.57}$	**78.20**$_{\pm 2.87}$
More than 10 classes	energy	50	$17.77_{\pm 6.15}$	N/A	$12.30_{\pm 2.59}$	$12.11_{\pm 3.16}$	$10.14_{\pm 2.87}$	$10.55_{\pm 2.44}$	$11.99_{\pm 2.27}$	$15.46_{\pm 3.54}$	N/A	**23.98**$_{\pm 2.73}$
		100	$25.94_{\pm 4.86}$	N/A	$17.78_{\pm 4.73}$	$18.60_{\pm 6.09}$	$18.56_{\pm 6.39}$	$18.84_{\pm 6.23}$	$19.91_{\pm 5.21}$	$17.65_{\pm 5.88}$	N/A	**31.24**$_{\pm 5.53}$
		200	$35.99_{\pm 8.92}$	N/A	$27.65_{\pm 11.12}$	$27.77_{\pm 10.55}$	$28.37_{\pm 10.82}$	$29.50_{\pm 10.33}$	$29.57_{\pm 9.18}$	$28.95_{\pm 10.40}$	N/A	**41.28**$_{\pm 7.66}$
	collins	100	$11.44_{\pm 2.77}$	N/A	$8.38_{\pm 1.52}$	$8.11_{\pm 1.00}$	$7.93_{\pm 1.40}$	$12.67_{\pm 2.16}$	$7.53_{\pm 1.10}$	$9.21_{\pm 2.35}$	N/A	**13.07**$_{\pm 2.51}$
		200	$15.74_{\pm 3.73}$	**17.45**$_{\pm 3.46}$	$12.08_{\pm 3.03}$	$11.37_{\pm 1.20}$	$10.74_{\pm 1.72}$	$15.39_{\pm 3.37}$	$10.71_{\pm 1.37}$	$14.30_{\pm 3.42}$	N/A	$17.03_{\pm 3.20}$
	texture	50	$72.40_{\pm 13.07}$	$76.40_{\pm 10.50}$	$55.32_{\pm 6.20}$	$54.80_{\pm 12.97}$	$55.39_{\pm 10.65}$	$62.27_{\pm 8.01}$	$55.65_{\pm 10.58}$	$62.94_{\pm 12.06}$	N/A	**78.90**$_{\pm 7.96}$
		100	$82.42_{\pm 10.38}$	$84.35_{\pm 9.67}$	$66.00_{\pm 7.21}$	$69.49_{\pm 10.93}$	$71.78_{\pm 9.06}$	$76.25_{\pm 7.40}$	$70.93_{\pm 9.71}$	$76.34_{\pm 9.55}$	N/A	**86.01**$_{\pm 7.36}$
		200	$87.54_{\pm 7.62}$	$89.29_{\pm 6.20}$	$78.37_{\pm 6.03}$	$82.44_{\pm 7.15}$	$81.94_{\pm 6.30}$	$84.67_{\pm 4.79}$	$83.29_{\pm 6.32}$	$82.53_{\pm 7.99}$	N/A	**89.77**$_{\pm 5.77}$
		500	$92.96_{\pm 4.07}$	$93.69_{\pm 3.83}$	$90.09_{\pm 3.56}$	$91.48_{\pm 3.50}$	$90.50_{\pm 2.71}$	$91.53_{\pm 3.29}$	$91.76_{\pm 3.98}$	$91.24_{\pm 3.56}$	N/A	**93.76**$_{\pm 3.64}$

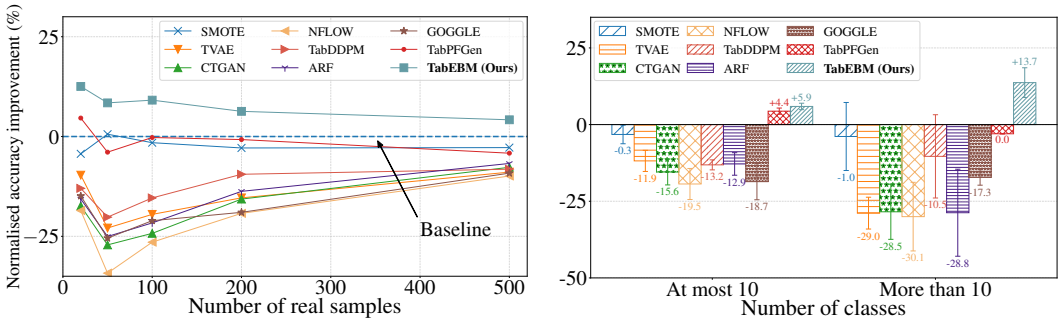


Figure 2: Mean normalised balanced accuracy improvement (%) across different sample sizes (**Left**) and across datasets with varying numbers of classes (**Right**). Positive values indicate that the generator improves downstream classification performance. TabEBM generally outperforms benchmark generators across varying sample sizes and number of classes.

▷ **Finding 1:** *TabEBM effectively improves downstream performance across sample sizes, especially for very low-sample-size regimes.* Table 1 and Figure 2 (Left) show that TabEBM is the only generator that consistently improves downstream performance across sample sizes.

▷ **Finding 2:** *TabEBM effectively improves downstream performance across the number of classes, especially for more than ten classes.* Figure 2 (Right) shows that TabEBM consistently outperforms the Baseline with notable improvements when the number of classes increases.

▷ **Finding 3:** *TabEBM is computationally efficient.* In Appendix E.4, we further discuss the trade-off between accuracy and the time needed for generating stratified synthetic data. The results show that TabEBM is practical, as it achieves higher downstream accuracy with lower time costs.

Discussion. We attribute the performance degradation in benchmark generators to their reliance on a single shared model to approximate all class-conditional densities. For instance, TabPFGen [40] leverages pre-trained Prior-Data Fitted Networks (PFNs), however, it shares a generator across all classes, which can lead to inaccurate density estimates (see examples in Appendix D). In contrast, TabEBM focuses on approximating one class at a time, free from the noise of other classes. Appendix A provides detailed discussions on the rationales of TabEBM's model design.

4 Conclusion

We introduce TabEBM, a novel tabular data augmentation method that creates class-specific EBM generators, learning the marginal distribution for each class separately. We also provide one of the first comprehensive analyses of tabular data augmentation across various dataset sizes. Our results demonstrate that TabEBM improves downstream performance through data augmentation on real-world datasets, outperforming other benchmark generators.

References

[1] Antreas Antoniou, Amos Storkey, and Harrison Edwards. Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*, 2017.

[2] Randall Balestriero, Leon Bottou, and Yann LeCun. The effects of regularization and data augmentation are class dependent. *Advances in Neural Information Processing Systems*, 35:37878–37891, 2022.

[3] Ms Aayushi Bansal, Dr Rewa Sharma, and Dr Mamta Kathuria. A systematic review on data scarcity problem in deep learning: solution and applications. *ACM Computing Surveys (CSUR)*, 54(10s):1–29, 2022.

[4] Andreas D Baxevanis, Gary D Bader, and David S Wishart. *Bioinformatics*. John Wiley & Sons, 2020.

[5] B. Bischl, Giuseppe Casalicchio, Matthias Feurer, Frank Hutter, Michel Lang, Rafael Gomes Mantovani, Jan N. van Rijn, and Joaquin Vanschoren. Openml benchmarking suites. *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS 2021)*, 2021.

[6] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[7] Vadim Borisov, Kathrin Sessler, Tobias Leemann, Martin Pawelczyk, and Gjergji Kasneci. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations*, 2022.

[8] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[9] Chenjing Cai, Shiwei Wang, Youjun Xu, Weilin Zhang, Ke Tang, Qi Ouyang, Luhua Lai, and Jianfeng Pei. Transfer learning for drug discovery. *Journal of Medicinal Chemistry*, 63(16):8683–8694, 2020.

[10] Rees Chang, Yu-Xiong Wang, and Elif Ertekin. Towards overcoming data scarcity in materials science: unifying models and datasets with a mixture of experts framework. *npj Computational Materials*, 8(1):242, 2022.

[11] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[12] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[13] David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 20(2):215–232, 1958.

[14] Jasper Dekoninck, Mark Niklas Müller, Maximilian Baader, Marc Fischer, and Martin Vechev. Evading data contamination detection for language models is (too) easy. *arXiv preprint arXiv:2402.02823*, 2024.

[15] Chunyuan Deng, Yilun Zhao, Xiangru Tang, Mark B. Gerstein, and Arman Cohan. Investigating data contamination in modern benchmarks for large language models. In *North American Chapter of the Association for Computational Linguistics*, 2024.

[16] Samuel Dooley, Gurnoor Singh Khurana, Chirag Mohapatra, Siddartha V Naidu, and Colin White. Forecastpfn: Synthetically-trained zero-shot forecasting. *Advances in Neural Information Processing Systems*, 36, 2024.

[17] Georgios Douzas and Fernando Bacao. Effective data generation for imbalanced learning using conditional generative adversarial networks. *Expert Systems with applications*, 91:464–471, 2018.

[18] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. *Advances in neural information processing systems*, 32, 2019.

[19] Xi Fang, Weijie Xu, Fiona Anting Tan, Ziqing Hu, Jiani Zhang, Yanjun Qi, Srinivasan H. Sengamedu, and Christos Faloutsos. Large language models (llms) on tabular data: Prediction, generation, and understanding - a survey. *Transactions on Machine Learning Research*, 2024.

[20] Steven Y Feng, Varun Gangal, Jason Wei, Sarath Chandar, Soroush Vosoughi, Teruko Mitamura, and Eduard Hovy. A survey of data augmentation approaches for nlp. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 968–988, 2021.

[21] Evelyn Fix. *Discriminatory analysis: nonparametric discrimination, consistency properties*, volume 1. USAF school of Aviation Medicine, 1985.

[22] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[23] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *International Conference on Learning Representations*, 2019.

[24] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520, 2022.

[25] Lasse Hansen, Nabeel Seedat, Mihaela van der Schaar, and Andrija Petrovic. Reimagining synthetic tabular data generation through data-centric ai: A comprehensive benchmark. *Advances in Neural Information Processing Systems*, 36:33781–33823, 2023.

[26] Mikel Hernandez, Gorka Epelde, Ane Alberdi, Rodrigo Cilla, and Debbie Rankin. Synthetic data generation for tabular health records: A systematic review. *Neurocomputing*, 493:28–45, 2022.

[27] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[29] Dipendra Jha, Kamal Choudhary, Francesca Tavazza, Wei-keng Liao, Alok Choudhary, Carelyn Campbell, and Ankit Agrawal. Enhancing materials property prediction by leveraging computational and experimental data using deep transfer learning. *Nature communications*, 10(1):5316, 2019.

[30] Minhao Jiang, Ken Ziyu Liu, Ming Zhong, Rylan Schaeffer, Siru Ouyang, Jiawei Han, and Sanmi Koyejo. Investigating data contamination for pre-training language models. *arXiv preprint arXiv:2401.06059*, 2024.

[31] Xiangjian Jiang, Andrei Margeloiu, Nikola Simidjievski, and Mateja Jamnik. Protogate: Prototype-based neural networks with global-to-local feature selection for tabular biomedical data. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

[32] Jayoung Kim, Chaejeong Lee, and Noseong Park. Stasy: Score-based tabular data synthesis. In *The Eleventh International Conference on Learning Representations*, 2022.

[33] Polina Kirichenko, Mark Ibrahim, Randall Balestriero, Diane Bouchacourt, Shanmukha Ramakrishna Vedantam, Hamed Firooz, and Andrew G Wilson. Understanding the detrimental class-level effects of data augmentation. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. Tabddpm: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pages 17564–17579. PMLR, 2023.

[35] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[36] Chaejeong Lee, Jayoung Kim, and Noseong Park. Codi: Co-evolving contrastive diffusion models for mixed-type tabular synthesis. In *International Conference on Machine Learning*, pages 18940–18956. PMLR, 2023.

[37] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[38] Roman Levin, Valeriia Cherepanova, Avi Schwarzschild, Arpit Bansal, C Bayan Bruss, Tom Goldstein, Andrew Gordon Wilson, and Micah Goldblum. Transfer learning with deep tabular models. In *The Eleventh International Conference on Learning Representations*, 2022.

[39] Tennison Liu, Zhaozhi Qian, Jeroen Berrevoets, and Mihaela van der Schaar. Goggle: Generative modelling for tabular data by learning relational structure. In *The Eleventh International Conference on Learning Representations*, 2023.

[40] Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. Tabpfgen–tabular data generation with tabpfn. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[41] Inbal Magar and Roy Schwartz. Data contamination: From memorization to exploitation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 157–165, 2022.

[42] Bradley A Malin, Khaled El Emam, and Christine M O'Keefe. Biomedical data privacy: problems, perspectives, and recent advances. *Journal of the American medical informatics association*, 20(1):2–6, 2013.

[43] Dionysis Manousakas and Sergül Aydöre. On the usefulness of synthetic tabular data generation. In *Data-centric Machine Learning Research (DMLR) Workshop at the 40th International Conference on Machine Learning (ICML)*, 2023.

[44] Andrei Margeloiu, Nikola Simidjievski, Pietro Lio, and Mateja Jamnik. Weight predictor network with feature selection for small sample tabular biomedical data. *AAAI Conference on Artificial Intelligence*, 2023.

[45] Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Advances in Neural Information Processing Systems*, 36, 2024.

[46] Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD explorations newsletter*, 3(1):27–32, 2001.

[47] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do bayesian inference. In *International Conference on Learning Representations*, 2022.

[48] Alhassan Mumuni and Fuseini Mumuni. Data augmentation: A comprehensive survey of modern approaches. *Array*, 16:100258, 2022.

[49] Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, pages 25660–25676. PMLR, 2023.

[50] Noseong Park, Mahmoud Mohammadi, Kshitij Gorde, Sushil Jajodia, Hongkyu Park, and Youngmin Kim. Data synthesis based on generative adversarial networks. *Proceedings of the VLDB Endowment*, 11(10), 2018.

[51] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[52] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[53] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

[54] Zhaozhi Qian, Rob Davis, and Mihaela van der Schaar. Synthcity: a benchmark framework for diverse use cases of tabular synthetic data. *Advances in Neural Information Processing Systems*, 36, 2024.

[55] Yiming Qin, Huangjie Zheng, Jiangchao Yao, Mingyuan Zhou, and Ya Zhang. Class-balancing diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18434–18443, 2023.

[56] Vignesh Sampath, Iñaki Maurtua, Juan Jose Aguilar Martin, and Aitor Gutierrez. A survey on generative adversarial networks for imbalance problems in computer vision tasks. *Journal of big Data*, 8:1–59, 2021.

[57] Nabeel Seedat, Nicolas Huynh, Boris van Breugel, and Mihaela van der Schaar. Curated llm: Synergy of llms and data curation for tabular augmentation in low-data regimes. In *Forty-first International Conference on Machine Learning*, 2024.

[58] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[59] Connor Shorten, Taghi M Khoshgoftaar, and Borko Furht. Text data augmentation for deep learning. *Journal of big Data*, 8(1):101, 2021.

[60] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

[61] Fahim Sufi. Addressing data scarcity in the medical domain: A gpt-based approach for synthetic data generation and feature extraction. *Information*, 15(5):264, 2024.

[62] Jordan Ubbens, Ian Stavness, and Andrew G Sharpe. Gpfn: Prior-data fitted networks for genomic prediction. *bioRxiv*, pages 2023–09, 2023.

[63] Boris Van Breugel, Zhaozhi Qian, and Mihaela Van Der Schaar. Synthetic data, real errors: how (not) to publish and use synthetic data. In *International Conference on Machine Learning*, pages 34793–34808. PMLR, 2023.

[64] Boris van Breugel and Mihaela van der Schaar. Why tabular foundation models should be a research priority. In *Forty-first International Conference on Machine Learning*, 2024.

[65] David A Van Dyk and Xiao-Li Meng. The art of data augmentation. *Journal of Computational and Graphical Statistics*, 10(1):1–50, 2001.

[66] David S Watson, Kristin Blesch, Jan Kapar, and Marvin N Wright. Adversarial random forests for density estimation and generative modeling. In *International Conference on Artificial Intelligence and Statistics*, pages 5357–5375. PMLR, 2023.

[67] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[68] Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional gan. *Advances in neural information processing systems*, 32, 2019.

[69] Hengrui Zhang, Jiani Zhang, Zhengyuan Shen, Balasubramaniam Srinivasan, Xiao Qin, Christos Faloutsos, Huzefa Rangwala, and George Karypis. Mixed-type tabular data synthesis with score-based diffusion in latent space. In *The Twelfth International Conference on Learning Representations*, 2023.

Appendix:
Augmenting Small-size Tabular Data with Class-Specific Energy-Based Models

Table of Contents

A Extended Related Work & Discussion

Section 3 shows that TabEBM efficiently generates high-quality data that can effectively improve the downstream performance via data augmentation. In Table 2, we further provide a summary of tabular data generative models analysed from three important perspectives: (i) *Training:* the type of distribution that the generators learn (crucial for preserving the original training label distribution), and the training costs associated with learning; (ii) *Generation:* do the generators employ class-specific models (reflecting their capability to capture unique features essential for label-invariant generation), and do models support stratified generation (crucial for effective data augmentation); (iii) *Practicability:* the scalability of the generators with respect to the number of classes (a common requirement in real-world multi-class tasks), and consistent downstream performance improvement across different class sizes.

Generative Models for Tabular Data. The common paradigm for tabular data generation is to adapt Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) [68, 50]. For instance, TableGAN employs a convolutional neural network to optimise the label quality [50], and TVAE is introduced in [68] as a variant of VAE for tabular data. However, these methods learn the joint distribution and thus cannot preserve the stratification of the original data (Appendix D). CTGAN [68] refines the generation to be class-conditional. The recent ARF [66] is an adversarial variant of random forest for density estimation, and GOGGLE [39] enhances VAE by learning relational structure with a Graph Neural Network (GNN). Some recent work focuses on generation with denoising diffusion models [34, 69, 32, 36]. For instance, TabDDPM [34] demonstrates that diffusion models can approximate typical distributions of tabular data. Although these class-conditional models can preserve the label distribution, they struggle to outperform Baseline and standard SMOTE in data augmentation [57, 40].

We attribute the performance degradation in current class-conditional models to their reliance on a single shared model to approximate all class-conditional densities. For instance, another promising generative approach uses pre-trained models like Prior-Data Fitted Networks (PFNs), and the recent TabPFGen [40] adapts such models into one shared class-conditional generator. However, TabPFGen's shared generator can lead to inaccurate density estimates, particularly in high-noise and class-imbalance situations (see examples in Appendix D). As noise increases, TabPFGen's inferred densities fluctuate significantly and diverge from the true data distributions. In contrast, TabEBM uses class-specific EBMs to model each class's marginal distributions, and the results in Appendix D reveal that our design choice reduces the impact of noise and data imbalance. TabEBM focuses on approximating and generating for one class at a time, remaining unaffected by noise from other classes. Overall, our results demonstrate that TabEBM consistently improves performance across different datasets and sample sizes, outperforming TabPFGen. Moreover, TabPFGen is limited in usability (e.g., it supports only up to ten classes), while TabEBM scales to any number of classes.

In a broader context, some recent work attempts to adapt Large Language Models (LLMs) for tabular data generation [19, 57, 7]. However, data contamination is an inherent issue with such LLM-based models [15, 30, 14, 41]. As the pre-training data is not typically open-source, these models can have unfair advantages in downstream tasks (i.e., the full real dataset, including the real test data, may have been used for pre-training). Therefore, in this paper, we focus on models without support from LLMs, thus avoiding potential biases from data contamination.

Table 2: **Comparison of the properties between TabEBM and prior tabular generative methods.** TabEBM has novel design rationales of training-free class-specific models, and TabEBM is highly practicable with wide applicability and consistent accuracy improvement.

Methods	Category	Training		Generation		Practicability		
		Learned distribution	Training-free	Class-specific models	Stratified generation	Unlimited classes	ACC improve ($\leq$ 10 classes)	ACC improve ($>$ 10 classes)
SMOTE [11]	Interpolation	N/A	✔	N/A	✔	✔	✗	✗
TVAE [68]	VAE	$p(\mathbf{x}, y)$	✗	✗	✗	✔	✗	✗
CTGAN [68]	GAN	$p(\mathbf{x} \mid y)$	✗	✗	✔	✔	✗	✗
NFLOW [18]	Normal. Flows	$p(\mathbf{x}, y)$	✗	✗	✗	✔	✗	✗
TabDDPM [34]	Diffusion	$p(\mathbf{x} \mid y)$	✗	✗	✔	✔	✗	✗
ARF [66]	Random Forest	$p(\mathbf{x}, y)$	✗	✗	✗	✔	✗	✗
GOGGLE [39]	GNN	$p(\mathbf{x} \mid y)$	✗	✗	✔	✔	✗	✗
TabPFGen [40]	PFN	$p(\mathbf{x} \mid y)$	✔	✗	✔	✗	✔	✗
TabEBM (Ours)	PFN	$p(\mathbf{x} \mid y)$	✔	✔	✔	✔	✔	✔

Data Augmentation (DA) for Tabular Data. DA is an omnipresent technique in computer vision and natural language processing [65, 59, 58, 48, 20, 1]. However, DA for tabular data remains underexplored, and existing methods often perform poorly in real-world tasks, sometimes even reducing performance [43]. Recent studies show that using the same transformations across all classes leads to varied performance impacts [2, 33], indicating that data augmentation effects are class-specific and suggesting that different classes may require distinct augmentations. Given the lack of symmetries in tabular data, we believe this class-dependent effect is even more pronounced. Therefore, we propose TabEBM as a class-specific generative model to produce tailored augmentations for each class.

Prior-fitted Networks (PFNs) for Tabular Data. Recent work proposes to approximate the posterior predictive distribution with transformers [47, 27, 49, 62, 16]. PFNs can be adapted for various purposes by pre-training the transformer with corresponding "prior data", and then it can make in-context predictions with unseen downstream data. For instance, TabPFN is a variant that is pre-trained on a prior designed for tabular data [27]. We note that prior data is different to synthetic data in this paper. Specifically, prior data refers to manually crafted fake data (e.g., $y = 2x$) with no real-world semantics. In contrast, synthetic data from generators is expected to have the same semantics as real data. Inspired by TabPFN's success in small-size classification tasks, TabEBM converts TabPFN into multiple EBMs that learn the marginal distribution for each class. The training-free nature of TabPFN enables TabEBM to generate high-quality tabular data without introducing extra training costs. Additionally, our class-specific design lets TabEBM surpass TabPFN's limits and scale to more than ten classes.

Limitations and Future Work. TabEBM is a general method that relies on an underlying binary classifier, and as such, its strengths and weaknesses are directly tied to this classifier. We used TabPFN because it is a well-established open-source pre-trained model for tabular data. Therefore, TabEBM inherits some of TabPFN's limitations, particularly in scaling to a larger number of features. TabEBM can handle datasets with over 1000 samples, overcoming TabPFN's limitation, as it processes one class at a time. Our results also show that TabEBM can handle categorical data by encoding categorical features with leave-one-out target statistics. We stress that TabEBM is compatible with any classifier that can be adapted into EBMs, as described in Section 2. As foundational models for tabular data evolve [64], new models capable of handling more features and samples are expected. Integrating them into TabEBM will enhance its ability to manage high-dimensional datasets, increasing its versatility and utility.

B Pseudocode for TabEBM Sampling

Algorithm 1 TabEBM sampling from Class-Specific EBM $E_c(\mathbf{x})$

Input: Training data $\mathcal{X}_c$ for class c, step size α_{step}, noise scale α_{noise}, initial perturbation σ_{start}, number of steps T
Output: Set of synthetic samples for class c

Initialise the surrogate binary classification task and train the model
Assign new labels to the samples $\mathcal{X}_c$ from class c, setting them to class 1
Generate a set of surrogate negative samples $\mathcal{X}_c^{\text{neg}}$ and assign them class 0 labels
Train a binary classifier f_θ^c on the dataset $\mathcal{D}_c = (\mathcal{X}_c \cup \mathcal{X}_c^{\text{neg}}, \{1\}^{|\mathcal{X}_c|} \cup \{0\}^{|\mathcal{X}_c^{\text{neg}}|})$
Synthesise samples using Stochastic Gradient Langevin Dynamics (SGLD)
Initialise synthetic data points $\mathbf{x}_0^{\text{synth}}$ by sampling from $\mathcal{N}(\mathcal{X}_c, \sigma_{\text{start}}^2 \mathbf{I})$
for each iteration $t = 0, 1, \ldots, T-1$ **do**
    $E_c(\mathbf{x}_t^{\text{synth}}) = -\log\left(\exp(f_\theta^c(\mathbf{x}_t^{\text{synth}})[0]) + \exp(f_\theta^c(\mathbf{x}_t^{\text{synth}})[1])\right)$
    $\mathbf{x}_{t+1}^{\text{synth}} = \mathbf{x}_t^{\text{synth}} - \alpha_{\text{step}} \nabla E_c(\mathbf{x}_t^{\text{synth}}) + \mathcal{N}(0, \alpha_{\text{noise}}^2 \mathbf{I})$
end for
return $\mathbf{x}_T^{\text{synth}}$ as the generated synthetic data for class c

C Reproducibility

C.1 Datasets

All eight datasets are publicly available on OpenML [5], and their details are listed in Table 3. To ensure consistent stratified data-splitting across all datasets, we remove classes with fewer than 10 sam-

ples. For example, the original "energy" dataset contains 14 classes with fewer than 10 samples, which could result in a validation set lacking samples from these classes, leading to unstratified data splitting.

Table 3: Details of the eight real-world tabular datasets.

Dataset	OpenML ID	Not evaluated in TabPFN [27]	# Samples (N)	# Features (D)	# Classes	N/D	# Samples per class (Min)	# Samples per class (Max)
At most 10 classes								
protein	40966	✔	1,080	77	8	14.03	105	150
fourier	14	✗	2,000	76	10	26.32	200	200
biodeg	1494	✗	1,055	41	2	25.73	356	699
steel	1504	✗	1,941	33	2	58.82	673	1,268
stock	841	✔	950	9	2	105.56	462	488
More than 10 classes								
energy	1472	✔	698	9	23	77.56	10	74
collins	40971	✔	970	19	26	51.05	17	80
texture	40499	✔	5,500	40	11	137.5	500	500

C.2 Data Splitting

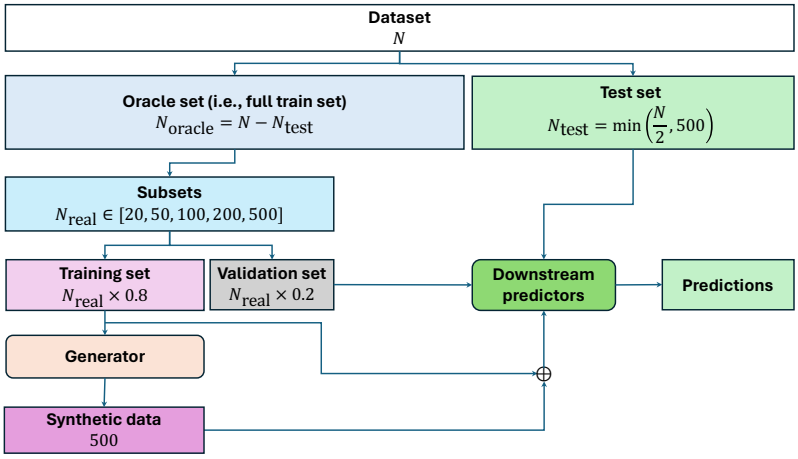


Figure 3: Data splitting strategies for data augmentation for all datasets.

C.3 Data Preprocessing

Following the procedures presented in prior work [45, 24], we perform preprocessing in two steps. We first compute the required statistics with training data and then transform it. Firstly, we impute the missing values with the mean value for numerical features and the most mode value for categorical features. Secondly, we convert the categorical features into numerical features equal to Leave-one-out Target Statistic [53, 46]. Next, we perform Z-score normalisation for each feature. Specifically, we compute each feature's mean and standard deviation in the training data and then transform the training samples to have a mean of zero and a variance of one for each feature. Finally, we apply the same transformation to the validation and test data before conducting evaluations.

C.4 Software and Computing Resources

Software implementation. *(i) For generators:* We implemented TabEBM using PyTorch 1.13 [51], an open-source deep learning library with a BSD licence. We implemented SMOTE with Imbalanced-learn [37], an open-source Python library for imbalanced datasets with an MIT licence. For other benchmark generators, we used their open-source implementations in Synthcity [**?**], a library for generating and evaluating synthetic tabular data with an Apache-2.0 license. *(ii) For downstream predictors:* We implemented TabPFN with its open-source implementation (`https://github.com/automl/TabPFN`). We implemented the other five downstream predictors (i.e., Logistic Regression, KNN, MLP, Random Forest and XGBoost) with their open-source implementation in scikit-learn [52],

an open-source Python library under the 3-Clause BSD license. *(iii) For result analysis and visualisation:* All numerical plots and graphics have been generated using Matplotlib 3.7 [28], a Python-based plotting library with a BSD licence. The model architecture was generated using draw.io (https://github.com/jgraph/drawio), a free drawing software under Apache License 2.0.

We ensure the consistency and reproducibility of experimental results by implementing a uniform pipeline using PyTorch Lightning, an open-source library under an Apache-2.0 licence. We further fixed the random seeds for data loading and evaluation throughout the training and evaluation process. This ensured that TabEBM and all benchmark models were trained and evaluated on the same set of samples. We also attach our code to this submission and will release it under the MIT licence upon publication. The experimental environment settings, including library dependencies, are specified in the associated code for reference and reproduction purposes.

Computing Resources. We trained 140,000 models for evaluations (including over 35,000 of generators and over 10,500 for downstream predictors). All our experiments are run on a single machine from an internal cluster with a GPU Nvidia Quadro RTX 8000 with 48GB memory and an Intel(R) Xeon(R) Gold 5218 CPU with 16 cores (at 2.30GHz). The operating system was Ubuntu 20.4.4 LTS.

C.5 Implementation of Generators

TabEBM. In all our experiments, the surrogate binary classifier in TabEBM is a pretrained in-context model, TabPFN [27], using the official model weights released by the authors (https://github.com/automl/TabPFN/raw/main/tabpfn/models_diff/prior_diff_real_checkpoint_n_0_epoch_42.cpkt). We use TabPFN with three ensembles. We use four surrogate negative samples, $\mathcal{X}_c^{\text{neg}}$, positioned at $\alpha_{\text{dist}}^{\text{neg}} = 5$ standard deviations from zero, in random corners of a hypercube in $\mathbb{R}^D$ (as explained in Section 2.2), distant from any real data. In Appendix E.1, we show that TabEBM is robust to the distribution of the negative samples.

We use SGLD [67] for sampling from TabEBM, where the starting points $\mathbf{x}_0^{\text{synth}}$ are initialised by adding Gaussian noise with zero mean and standard deviation $\sigma_{\text{start}} = 0.01$ to a randomly selected sample of the specific class, i.e., $\mathbf{x}_0^{\text{synth}} \sim \mathcal{N}(\mathcal{X}_c, \sigma_{\text{start}}^2 \mathbf{I})$. For SGLD, we used the following parameters: step size $\alpha_{\text{step}} = 0.1$, noise scale $\alpha_{\text{noise}} = 0.01$ and number of steps $T = 200$. We found TabEBM to be robust to the SGLD settings (see Appendix E.2). We will release it as a public open-source library available after publication.

TabPFGen. We re-implemented TabPFGen [40] by closely following the original paper since no official implementation is available. As recommended in [40], the starting points are initialized by adding Gaussian noise with zero mean and standard deviation of 0.01 to the training points.

SMOTE. We use the open-source implementation of SMOTE from Imbalanced-learn [37], and the number neighbours k is set within the range of $\{1, 3, 5\}$. When applicable, we always set the maximum value for nearest neighbours (i.e., $k = 5$). However, very low-sample-size datasets may not contain sufficient samples for large k. For instance, the "fourier" dataset ($N_{\text{real}} = 20$) only has two samples per class. We set $k = 1$ to generate synthetic data with SMOTE in these cases.

For the other six benchmark generators, we use their open-source implementations in Synthcity [54]. Following prior studies [69, 63, 57, 40], we use the default settings for all generators.

C.6 Implementation of Downstream Predictors

We implemented TabPFN with its official implementation [27] and the other five downstream predictors with the scikit-learn library [52]. Following prior studies [63, 57], we use the default settings for all downstream predictors.

D Limitations of Existing Generative Methods

We showcase three limitations of current generative models: (1) Figure 4 shows that models approximating the joint distribution $p(\mathbf{x}, y)$ may fail to preserve the stratification of the real data and even fail to generate samples from specific classes. (2) Figure 5 evaluates the approximated class-conditional distributions $p(\mathbf{x} \mid y)$ on data with increasing noise levels, and (3) Figure 6 evaluates the approximated class-conditional distributions $p(\mathbf{x} \mid y)$ on data with increasing class imbalance.

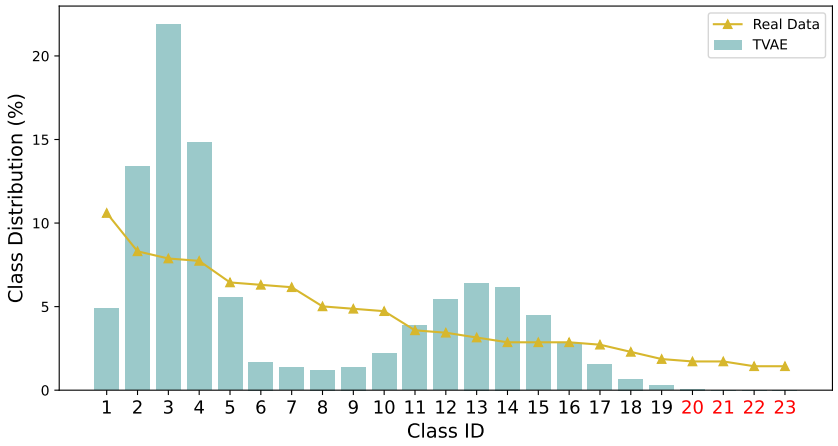
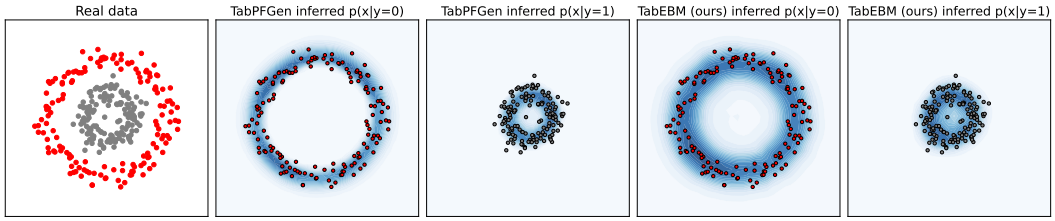
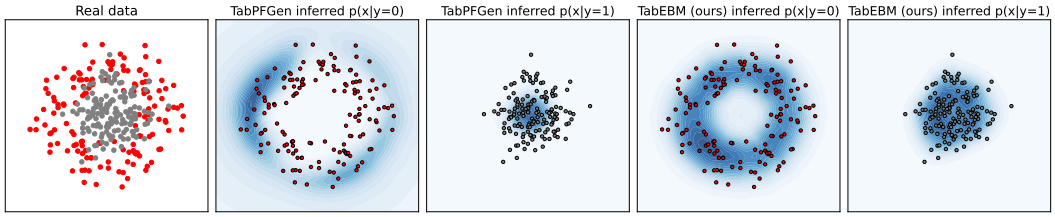


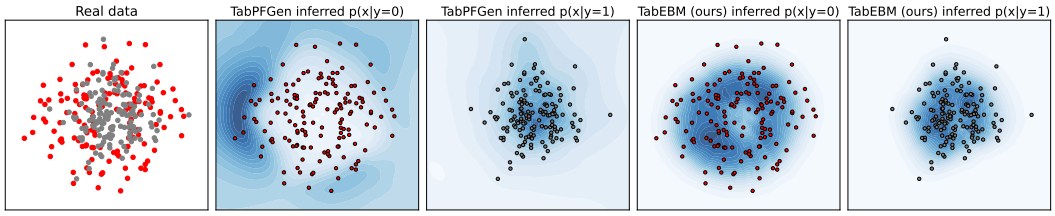
Figure 4: Comparison of class distribution between real data and synthetic data from TVAE. We first train TVAE on the "energy-efficiency" dataset and then randomly generate 10,000 samples with it. We highlight the classes where no synthetic samples are generated. TVAE fails to generate samples for 4 of 23 classes, showing the impracticability to preserve stratification by generative methods that learn joint distribution $p(\mathbf{x}, y)$.



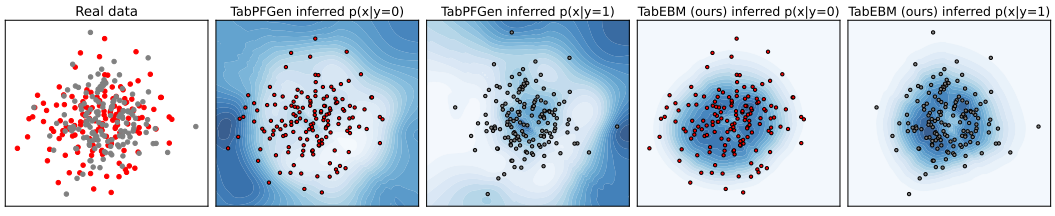
(a) Noise level 0.1



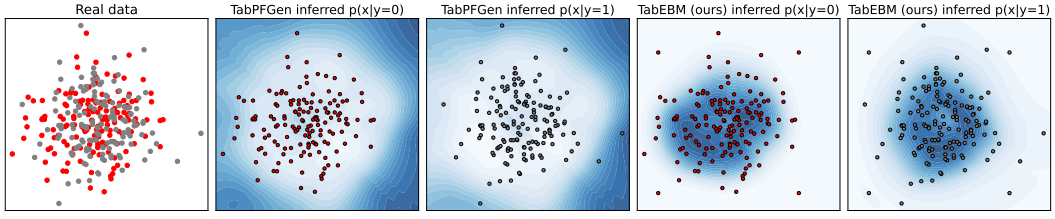
(b) Noise level 0.25



(c) Noise level 0.5

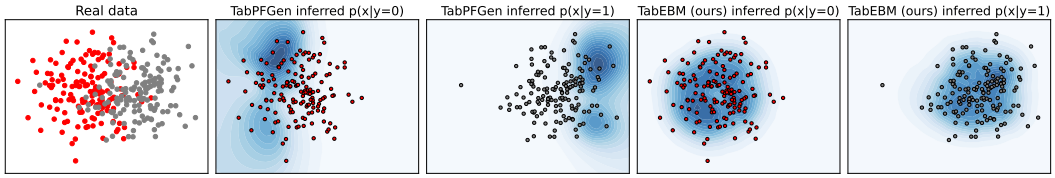


(d) Noise level 1

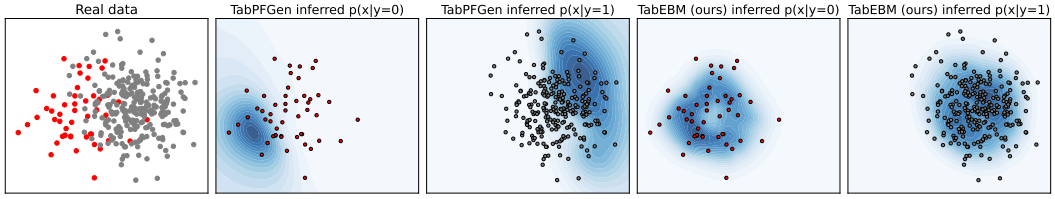


(e) Noise level 2

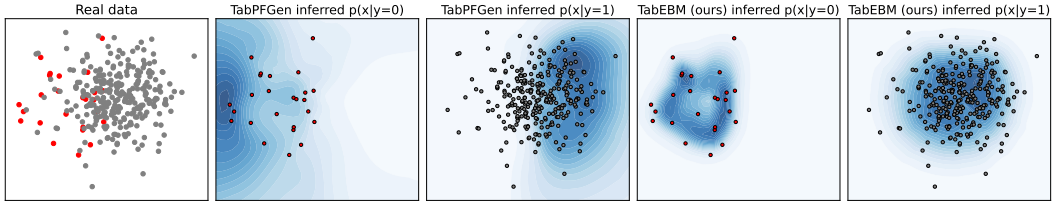
Figure 5: Evaluating the approximated class-conditional distributions on data with increasing noise levels. Darker blue indicates a higher assigned probability. TabPFGen uses a single shared energy-based model to infer the class-conditional distribution $p(\mathbf{x}|y)$. As noise increases, TabPFGen's probability assignments vary significantly and end up assigning very high probabilities that are far from the real data. For instance, the areas of assigned probability for $p(\mathbf{x}|y = 1)$ completely flip when noise increases from 0.5 to 1. In contrast, our TabEBM uses class-specific energy models, resulting in robust inferred conditionals. TabEBM performs well even under very high noise (see $p(\mathbf{x}|y = 0)$ for noise level 2), while TabPFGen struggles.



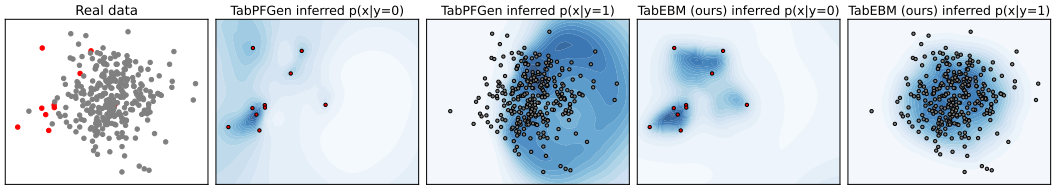
(a) Class ratio 150:150



(b) Class ratio 50:250



(c) Class ratio 25:275



(d) Class ratio 10:290

Figure 6: Evaluating the approximated class-conditional distributions on a toy dataset of 300 samples with varying class imbalances. The two clusters maintain their positions. Darker blue indicates a higher assigned probability. TabPFGen uses a single shared energy-based model to infer the class-conditional distribution $p(\mathbf{x}|y)$. As class imbalance increases, TabPFGen starts assigning high probability in areas far from the real data, for instance, in the case of $p(\mathbf{x}|y = 1)$ for class ratio 10:290. In contrast, our TabEBM fits class-specific energy models only on the class-wise data $\mathcal{X}_c = \{\mathbf{x}^{(i)} \mid y_i = c\}$. This results in very robust inferred conditional distributions even under heavy class imbalance (e.g., see that $p(\mathbf{x}|y = 1)$ remains relatively constant).

E Extended Experimental Results

E.1 Ablations on the distribution of the surrogate negative samples

E.1.1 Example of negative samples in 2D plane

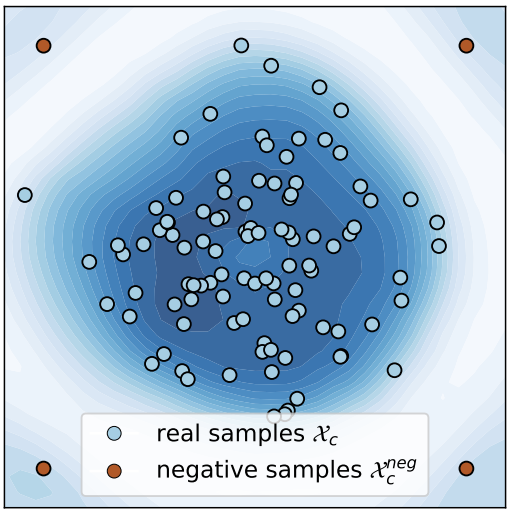


Figure 7: The class-specific energy function $E_c(\mathbf{x})$ from the surrogate binary task, where the blue region represents low energy (i.e., high data density). Placing the negative samples in a hypercube distant from the data results in an accurate energy function.

E.1.2 Ablations on placing the negative samples

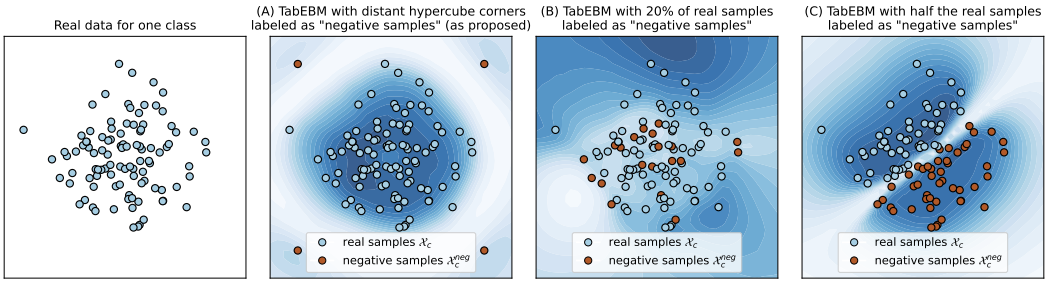


Figure 8: TabEBM energy $E_c(x)$ for different choices of negative samples. The blue region represents low energy, indicating high data density. In (A), TabEBM, with the proposed negative samples placed in a hypercube far from the data, infers an accurate energy surface, resulting in generated data close to the real points. In (B), labelling a random subset of the real data as negative samples leads to a completely inaccurate energy surface. In (C), labelling half of the real points as negative samples reduces density near the decision boundary, as TabPFN assigns low maximal logit due to the high uncertainty. In conclusion, placing negative samples far from the real data results in a robust energy surface.

Figure 8 shows TabEBM's energy $E_c(x)$ when varying the selection of the negative samples. TabEBM infers an accurate energy surface with distant negative samples, and the energy surface becomes inaccurate when negative samples resemble real samples. This occurs because TabPFN is uncertain when points of different classes are close, affecting its logits magnitude and making them unsuitable for density estimation.

E.1.3 Varying the number of negative samples

We evaluate the impact of the ratio $|\mathcal{X}_c^{\text{neg}}| : |\mathcal{X}_c|$ between the negative samples $\mathcal{X}_c^{\text{neg}}$ and the real samples $|\mathcal{X}_c|$. We vary $|\mathcal{X}_c^{\text{neg}}|$ while keeping $|\mathcal{X}_c|$ fixed, simulating both balanced and highly imbalanced scenarios. The negative samples are placed in random corners of the hypercube (as described in Section 2), at five standard deviations in each direction (i.e., $\alpha_{\text{dist}}^{\text{neg}} = 5$). To ensure reliable outcomes, we maintained a consistent ratio across all classes, keeping the same proportion of negative samples for each class.

Table 4 shows the results across six datasets with $N_{\text{real}} = 100$ real samples, demonstrating that TabEBM is robust to imbalances in the surrogate binary tasks. The column with $|\mathcal{X}_c^{\text{neg}}| = 4$ represents the TabEBM results from the main paper, where four negative samples were placed in the corners (as described in Section 2). There are negligible differences in performance, and TabEBM consistently outperforms both the baseline and other generators (as shown in Table 1).

Table 4: Evaluating the impact of varying the ratio $|\mathcal{X}_c^{\text{neg}}| : |\mathcal{X}_c|$. We show the test classification accuracy performance (%) of TabEBM on data augmentation averaged over six datasets and ten repeats. TabEBM shows consistent performance and outperforms the baseline, regardless of the number of negative samples.

	TabEBM					Baseline (Real data)
Ratio $\|\mathcal{X}_c^{\text{neg}}\| : \|\mathcal{X}_c\|$	0.1	0.2	0.5	1	Fixed $\|\mathcal{X}_c^{\text{neg}}\| = 4$	-
biodeg	$76.59_{\pm 3.95}$	$76.54_{\pm 3.95}$	$76.47_{\pm 4.05}$	$76.81_{\pm 3.58}$	$76.45_{\pm 3.08}$	$76.69_{\pm 2.70}$
steel	$92.71_{\pm 7.46}$	$92.60_{\pm 7.45}$	$92.79_{\pm 7.50}$	$92.63_{\pm 7.59}$	$92.71_{\pm 7.57}$	$86.87_{\pm 12.4}$
stock	$90.46_{\pm 3.49}$	$90.41_{\pm 3.65}$	$90.52_{\pm 3.52}$	$90.31_{\pm 3.63}$	$90.36_{\pm 3.14}$	$89.07_{\pm 3.71}$
energy	$31.20_{\pm 6.22}$	$31.20_{\pm 6.22}$	$30.89_{\pm 5.83}$	$30.90_{\pm 6.09}$	$31.24_{\pm 5.53}$	$25.94_{\pm 4.86}$
collins	$13.06_{\pm 2.88}$	$13.02_{\pm 2.85}$	$13.05_{\pm 2.89}$	$12.97_{\pm 2.79}$	$13.07_{\pm 2.51}$	$11.44_{\pm 2.77}$
texture	$85.91_{\pm 6.92}$	$85.91_{\pm 6.92}$	$85.94_{\pm 6.76}$	$86.26_{\pm 6.72}$	$86.01_{\pm 7.36}$	$82.42_{\pm 10.38}$
Average accuracy	64.99	64.95	64.94	64.98	64.97	62.07

E.1.4 Varying the distance of the negative samples

We assess the effect of varying the distance of negative samples. We use TabEBM with four negative samples positioned randomly at the corners of the hypercube, as outlined in Section 2 (this corresponds to the experimental setup from the main paper). The distance of the negative samples, denoted as $\alpha_{\text{dist}}^{\text{neg}}$, is varied. Table 5 demonstrates that TabEBM remains generally robust to changes in this distance, with only small performance variations across different datasets. Importantly, using TabEBM for data augmentation consistently improves performance by approximately 3% compared to the Baseline, regardless of the distance used.

Table 5: Evaluating the impact of varying the distance of the negative samples $\alpha_{\text{dist}}^{\text{neg}}$ across various datasets. We show the test classification accuracy performance (%) of TabEBM on data augmentation averaged over six datasets and ten repeats. TabEBM is robust, and optional tuning of the negative samples could slightly improve performance.

	TabEBM						Baseline (Real data)
Per-dimension distance α_d **of the negative samples**	0.1	0.2	0.5	1	2	5	-
biodeg	$76.72_{\pm 3.33}$	$76.62_{\pm 3.40}$	$77.12_{\pm 2.60}$	$76.85_{\pm 3.14}$	$76.50_{\pm 3.93}$	$76.45_{\pm 3.08}$	$76.69_{\pm 2.70}$
steel	$93.97_{\pm 5.76}$	$93.46_{\pm 6.24}$	$93.00_{\pm 6.92}$	$92.60_{\pm 7.31}$	$92.68_{\pm 7.38}$	$92.71_{\pm 7.57}$	$86.87_{\pm 12.4}$
stock	$90.42_{\pm 3.46}$	$90.29_{\pm 3.61}$	$90.56_{\pm 3.46}$	$90.38_{\pm 3.64}$	$90.43_{\pm 3.56}$	$90.36_{\pm 3.14}$	$89.07_{\pm 3.71}$
energy	$31.73_{\pm 6.21}$	$31.42_{\pm 6.08}$	$31.86_{\pm 6.12}$	$32.53_{\pm 5.96}$	$31.65_{\pm 6.06}$	$31.24_{\pm 5.53}$	$25.94_{\pm 4.86}$
collins	$13.03_{\pm 2.59}$	$12.92_{\pm 2.60}$	$12.97_{\pm 2.69}$	$13.03_{\pm 2.84}$	$13.08_{\pm 2.93}$	$13.07_{\pm 2.51}$	$11.44_{\pm 2.77}$
texture	$85.62_{\pm 7.41}$	$85.58_{\pm 7.49}$	$85.50_{\pm 7.65}$	$85.05_{\pm 8.21}$	$85.20_{\pm 7.95}$	$86.01_{\pm 7.36}$	$82.42_{\pm 10.38}$
Average accuracy	65.25	65.05	65.17	65.07	64.92	64.97	62.07

E.2 Ablations on the sensitivity to the hyperparameters of SGLD sampling

We vary two key hyperparameters of SGLD on the "biodeg" binary dataset with $N_{\text{real}} = 100$: the step size α_{step} and the noise scale α_{noise}. Table 6 shows that TabEBM remains stable with respect to these hyperparameters. Note that smaller values of α_{noise} are expected to perform better because SGLD sampling adds noise at each iteration (see Line 7 in Algorithm 1), thus larger values of α_{noise} will hinder convergence of the SGLD sampler.

Table 6: Test classification accuracy (%) of TabEBM (averaged over six downstream predictors) with different SGLD settings. Increasing α_{noise} (added at each SGLD step) is expected to degrade performance, as it causes the sampling to diverge further from the real data.

α_{noise}	α_{step}			
	0.1	0.3	0.5	1.0
0.01	76.45	**77.09**	77.04	76.58
0.02	76.86	76.96	76.77	76.26
0.05	75.93	75.89	75.94	**75.70**

E.3 Results on data augmentation

E.3.1 Aggregated results on eight OpenML datasets.

Table 7: **Classification accuracy** (%) aggregated over six downstream predictors, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean $\pm$ std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. Our method, TabEBM, consistently outperforms training on real data alone, and achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	$28.14_{\pm 6.83}$	N/A	$21.18_{\pm 1.48}$	$22.00_{\pm 3.43}$	$21.30_{\pm 2.84}$	$22.12_{\pm 5.30}$	$24.82_{\pm 2.88}$	$22.40_{\pm 9.28}$	$33.25_{\pm 5.01}$	**33.84**$_{\pm 4.92}$
		50	$50.72_{\pm 10.53}$	$54.52_{\pm 8.59}$	$39.54_{\pm 5.19}$	$36.32_{\pm 7.17}$	$35.37_{\pm 8.00}$	$35.11_{\pm 11.78}$	$41.99_{\pm 5.24}$	$37.53_{\pm 14.72}$	$54.45_{\pm 7.96}$	**55.91**$_{\pm 6.41}$
		100	$67.83_{\pm 11.72}$	$73.25_{\pm 7.48}$	$59.28_{\pm 7.20}$	$57.64_{\pm 9.95}$	$52.57_{\pm 9.55}$	$56.37_{\pm 9.64}$	$57.01_{\pm 8.56}$	$51.69_{\pm 16.68}$	$71.53_{\pm 9.87}$	**73.31**$_{\pm 6.77}$
		200	$81.66_{\pm 10.18}$	$85.65_{\pm 6.24}$	$76.42_{\pm 7.71}$	$74.88_{\pm 8.20}$	$72.10_{\pm 10.04}$	$75.86_{\pm 9.30}$	$74.07_{\pm 8.74}$	$73.57_{\pm 6.74}$	$84.95_{\pm 7.47}$	**86.14**$_{\pm 5.50}$
		500	$93.49_{\pm 5.28}$	$94.73_{\pm 3.67}$	$92.24_{\pm 3.73}$	$91.48_{\pm 4.43}$	$90.44_{\pm 5.54}$	$90.62_{\pm 5.63}$	$91.79_{\pm 4.53}$	$91.31_{\pm 5.20}$	$94.87_{\pm 3.70}$	**95.18**$_{\pm 3.10}$
	fourier	20	$28.30_{\pm 12.09}$	N/A	$21.32_{\pm 4.06}$	$18.19_{\pm 3.90}$	$17.30_{\pm 3.03}$	$15.35_{\pm 3.26}$	$21.75_{\pm 2.76}$	$16.70_{\pm 2.91}$	$36.72_{\pm 7.30}$	**37.13**$_{\pm 6.01}$
		50	$53.69_{\pm 8.04}$	$55.51_{\pm 7.43}$	$37.96_{\pm 4.48}$	$35.09_{\pm 7.46}$	$31.94_{\pm 8.99}$	$35.99_{\pm 13.06}$	$40.32_{\pm 6.70}$	$33.56_{\pm 14.02}$	$55.11_{\pm 10.66}$	**56.57**$_{\pm 7.12}$
		100	$63.70_{\pm 6.76}$	$64.10_{\pm 6.89}$	$50.46_{\pm 8.61}$	$49.26_{\pm 9.15}$	$44.58_{\pm 8.40}$	$52.79_{\pm 10.04}$	$51.13_{\pm 6.35}$	$41.93_{\pm 15.60}$	$63.86_{\pm 7.76}$	**65.21**$_{\pm 6.42}$
		200	$70.99_{\pm 4.88}$	$71.43_{\pm 4.47}$	$62.17_{\pm 7.29}$	$62.92_{\pm 7.87}$	$59.15_{\pm 8.33}$	$68.05_{\pm 6.91}$	$62.53_{\pm 6.97}$	$56.44_{\pm 10.13}$	$71.81_{\pm 5.35}$	**72.36**$_{\pm 3.77}$
		500	$77.72_{\pm 2.36}$	$77.51_{\pm 2.60}$	$73.29_{\pm 4.97}$	$74.61_{\pm 4.89}$	$71.74_{\pm 6.54}$	$77.04_{\pm 3.64}$	$74.31_{\pm 4.40}$	$70.61_{\pm 6.01}$	$77.15_{\pm 2.57}$	**78.20**$_{\pm 2.87}$
	biodeg	20	$66.20_{\pm 4.26}$	$68.59_{\pm 1.17}$	$66.77_{\pm 2.64}$	$58.03_{\pm 2.47}$	$59.37_{\pm 1.74}$	$52.72_{\pm 2.38}$	$61.17_{\pm 2.00}$	$61.39_{\pm 6.39}$	$68.99_{\pm 2.54}$	**69.79**$_{\pm 2.15}$
		50	$72.66_{\pm 3.98}$	$72.80_{\pm 3.08}$	$71.31_{\pm 2.71}$	$67.99_{\pm 3.63}$	$62.40_{\pm 4.28}$	$60.72_{\pm 10.11}$	$71.62_{\pm 2.43}$	$66.68_{\pm 6.00}$	$73.29_{\pm 3.53}$	**73.78**$_{\pm 3.42}$
		100	**76.69**$_{\pm 2.70}$	$76.31_{\pm 2.42}$	$75.38_{\pm 2.06}$	$74.82_{\pm 2.89}$	$69.50_{\pm 4.59}$	$68.28_{\pm 9.54}$	$74.42_{\pm 2.38}$	$71.68_{\pm 3.72}$	$76.22_{\pm 2.31}$	$76.45_{\pm 3.08}$
		200	$80.01_{\pm 2.66}$	$79.67_{\pm 2.56}$	$78.11_{\pm 2.68}$	$78.19_{\pm 1.78}$	$75.05_{\pm 4.68}$	$74.43_{\pm 8.09}$	$77.97_{\pm 2.32}$	$77.13_{\pm 3.01}$	$79.76_{\pm 2.63}$	**80.11**$_{\pm 2.33}$
		500	$82.63_{\pm 2.43}$	**82.85**$_{\pm 1.93}$	$82.13_{\pm 1.94}$	$82.42_{\pm 1.58}$	$81.11_{\pm 3.23}$	$79.19_{\pm 6.60}$	$81.92_{\pm 2.28}$	$81.24_{\pm 2.30}$	$82.35_{\pm 2.21}$	$82.29_{\pm 2.15}$
	steel	20	$57.51_{\pm 4.58}$	$58.32_{\pm 3.27}$	$57.99_{\pm 3.06}$	$56.61_{\pm 1.70}$	$53.89_{\pm 1.73}$	$55.74_{\pm 6.02}$	$54.24_{\pm 2.08}$	$53.04_{\pm 2.36}$	$63.21_{\pm 5.86}$	**63.27**$_{\pm 5.45}$
		50	$75.06_{\pm 10.43}$	$65.63_{\pm 4.00}$	$64.18_{\pm 3.95}$	$63.70_{\pm 6.10}$	$58.90_{\pm 6.39}$	$65.85_{\pm 14.84}$	$61.72_{\pm 3.39}$	$56.72_{\pm 3.47}$	$78.67_{\pm 11.79}$	**80.50**$_{\pm 8.67}$
		100	$86.87_{\pm 12.49}$	$74.61_{\pm 5.99}$	$70.12_{\pm 5.76}$	$69.89_{\pm 5.58}$	$65.67_{\pm 9.10}$	$76.01_{\pm 17.54}$	$67.33_{\pm 5.15}$	$60.56_{\pm 5.37}$	$90.58_{\pm 9.50}$	**92.71**$_{\pm 7.57}$
		200	$92.90_{\pm 9.14}$	$81.97_{\pm 4.12}$	$78.73_{\pm 5.06}$	$78.36_{\pm 6.98}$	$75.90_{\pm 9.57}$	$85.45_{\pm 15.03}$	$78.65_{\pm 6.70}$	$68.20_{\pm 5.30}$	$95.56_{\pm 5.85}$	**96.29**$_{\pm 4.64}$
		500	$97.52_{\pm 3.76}$	$92.44_{\pm 4.46}$	$92.47_{\pm 3.66}$	$92.42_{\pm 4.76}$	$88.20_{\pm 8.36}$	$96.34_{\pm 4.67}$	$90.41_{\pm 5.35}$	$84.23_{\pm 10.90}$	$98.14_{\pm 2.67}$	**98.47**$_{\pm 2.15}$
	stock	20	$78.75_{\pm 4.39}$	$82.18_{\pm 2.15}$	$74.11_{\pm 3.71}$	$64.25_{\pm 6.29}$	$72.64_{\pm 2.01}$	$78.61_{\pm 3.57}$	$69.54_{\pm 1.65}$	$76.35_{\pm 5.08}$	$82.42_{\pm 2.17}$	**83.49**$_{\pm 1.60}$
		50	$86.10_{\pm 3.62}$	$87.82_{\pm 3.41}$	$82.81_{\pm 3.51}$	$79.63_{\pm 3.93}$	$80.14_{\pm 3.90}$	$86.72_{\pm 4.29}$	$82.48_{\pm 2.95}$	$83.36_{\pm 5.23}$	$88.14_{\pm 3.01}$	**88.44**$_{\pm 3.14}$
		100	$89.07_{\pm 3.71}$	$89.99_{\pm 3.22}$	$87.55_{\pm 4.25}$	$86.44_{\pm 4.40}$	$84.64_{\pm 4.79}$	$89.40_{\pm 4.26}$	$87.32_{\pm 4.42}$	$87.44_{\pm 5.46}$	$90.27_{\pm 3.33}$	**90.36**$_{\pm 3.51}$
		200	$90.85_{\pm 4.39}$	**91.75**$_{\pm 3.73}$	$90.12_{\pm 5.44}$	$89.44_{\pm 4.94}$	$88.47_{\pm 6.06}$	$90.76_{\pm 5.27}$	$89.59_{\pm 5.37}$	$89.62_{\pm 6.29}$	$91.56_{\pm 3.91}$	$91.71_{\pm 3.77}$
More than 10 classes	energy	50	$17.77_{\pm 6.15}$	N/A	$12.30_{\pm 2.59}$	$12.11_{\pm 3.16}$	$10.14_{\pm 2.87}$	$10.55_{\pm 2.44}$	$11.99_{\pm 2.27}$	$15.46_{\pm 3.54}$	N/A	**23.98**$_{\pm 2.73}$
		100	$25.94_{\pm 4.86}$	N/A	$17.78_{\pm 4.73}$	$18.60_{\pm 6.09}$	$18.56_{\pm 6.39}$	$18.84_{\pm 6.23}$	$19.91_{\pm 5.21}$	$17.65_{\pm 5.88}$	N/A	**31.24**$_{\pm 5.53}$
		200	$35.99_{\pm 8.92}$	N/A	$27.65_{\pm 11.12}$	$27.77_{\pm 10.55}$	$28.37_{\pm 10.82}$	$29.50_{\pm 10.33}$	$29.57_{\pm 9.18}$	$28.95_{\pm 10.40}$	N/A	**41.28**$_{\pm 7.66}$
	collins	100	$11.44_{\pm 2.77}$	N/A	$8.38_{\pm 1.52}$	$8.11_{\pm 1.00}$	$7.93_{\pm 1.40}$	$12.67_{\pm 2.16}$	$7.53_{\pm 1.10}$	$9.21_{\pm 2.35}$	N/A	**13.07**$_{\pm 2.51}$
		200	$15.74_{\pm 3.73}$	**17.45**$_{\pm 3.46}$	$12.08_{\pm 3.03}$	$11.37_{\pm 1.20}$	$10.74_{\pm 1.72}$	$15.39_{\pm 3.37}$	$10.71_{\pm 1.37}$	$14.30_{\pm 3.42}$	N/A	$17.03_{\pm 3.20}$
	texture	50	$72.40_{\pm 13.07}$	$76.40_{\pm 10.50}$	$55.32_{\pm 6.20}$	$54.80_{\pm 12.97}$	$55.39_{\pm 10.65}$	$62.27_{\pm 8.01}$	$55.65_{\pm 10.58}$	$62.94_{\pm 12.06}$	N/A	**78.90**$_{\pm 7.96}$
		100	$82.42_{\pm 10.38}$	$84.35_{\pm 9.67}$	$66.00_{\pm 7.21}$	$69.49_{\pm 10.93}$	$71.78_{\pm 9.06}$	$76.25_{\pm 7.40}$	$70.93_{\pm 9.71}$	$76.34_{\pm 9.55}$	N/A	**86.01**$_{\pm 7.36}$
		200	$87.54_{\pm 7.62}$	$89.29_{\pm 6.20}$	$78.37_{\pm 6.03}$	$82.44_{\pm 7.15}$	$81.94_{\pm 6.30}$	$84.67_{\pm 4.79}$	$83.29_{\pm 6.32}$	$82.53_{\pm 7.99}$	N/A	**89.77**$_{\pm 5.77}$
		500	$92.96_{\pm 4.07}$	$93.69_{\pm 3.83}$	$90.09_{\pm 3.56}$	$91.48_{\pm 3.50}$	$90.50_{\pm 2.71}$	$91.53_{\pm 3.29}$	$91.76_{\pm 3.98}$	$91.24_{\pm 3.56}$	N/A	**93.76**$_{\pm 3.64}$
Average rank			$3.30_{\pm 1.02}$	$3.03_{\pm 1.25}$	$6.79_{\pm 1.80}$	$7.48_{\pm 1.50}$	$8.94_{\pm 0.70}$	$6.39_{\pm 2.41}$	$6.94_{\pm 1.50}$	$7.76_{\pm 2.03}$	$3.15_{\pm 1.27}$	**1.21**$_{\pm 0.74}$

E.3.2 Predictor-wise results on eight OpenML datasets.

Table 8: **Classification accuracy** (%) of LR, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean ± std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable or the downstream predictor failed to converge, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. TabEBM achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	36.33$_{\pm 3.04}$	N/A	22.02$_{\pm 2.91}$	21.04$_{\pm 4.76}$	18.40$_{\pm 4.82}$	18.77$_{\pm 3.84}$	25.92$_{\pm 4.30}$	36.61$_{\pm 2.53}$	**38.07**$_{\pm 1.25}$	38.01$_{\pm 2.38}$
		50	62.14$_{\pm 3.77}$	61.43$_{\pm 4.34}$	37.04$_{\pm 2.79}$	33.10$_{\pm 5.99}$	31.25$_{\pm 4.21}$	23.98$_{\pm 2.75}$	43.64$_{\pm 5.07}$	54.95$_{\pm 3.28}$	63.00$_{\pm 3.69}$	**63.05**$_{\pm 3.84}$
		100	79.97$_{\pm 3.24}$	79.53$_{\pm 3.37}$	61.07$_{\pm 5.06}$	55.44$_{\pm 1.92}$	46.37$_{\pm 4.10}$	45.55$_{\pm 4.24}$	56.77$_{\pm 3.06}$	67.25$_{\pm 4.50}$	**80.54**$_{\pm 3.27}$	80.32$_{\pm 3.12}$
		200	91.53$_{\pm 1.58}$	90.92$_{\pm 1.81}$	77.43$_{\pm 2.75}$	71.27$_{\pm 3.07}$	66.16$_{\pm 4.31}$	66.37$_{\pm 3.42}$	70.52$_{\pm 2.17}$	76.30$_{\pm 3.70}$	**91.69**$_{\pm 1.66}$	91.34$_{\pm 1.77}$
		500	97.86$_{\pm 0.83}$	97.69$_{\pm 0.80}$	90.77$_{\pm 0.93}$	89.05$_{\pm 1.52}$	85.09$_{\pm 1.99}$	83.58$_{\pm 2.22}$	88.55$_{\pm 1.54}$	90.64$_{\pm 0.81}$	**97.97**$_{\pm 0.61}$	97.88$_{\pm 0.86}$
	fourier	20	42.90$_{\pm 5.30}$	N/A	22.46$_{\pm 5.88}$	16.00$_{\pm 4.70}$	15.48$_{\pm 3.79}$	13.58$_{\pm 4.30}$	22.04$_{\pm 4.42}$	15.80$_{\pm 4.15}$	**44.67**$_{\pm 8.85}$	43.02$_{\pm 5.14}$
		50	**60.62**$_{\pm 1.64}$	58.40$_{\pm 1.95}$	33.42$_{\pm 2.98}$	31.18$_{\pm 5.47}$	28.70$_{\pm 3.74}$	26.18$_{\pm 3.80}$	39.04$_{\pm 3.12}$	40.00$_{\pm 4.97}$	60.07$_{\pm 2.14}$	60.36$_{\pm 1.55}$
		100	**67.76**$_{\pm 2.49}$	65.84$_{\pm 2.35}$	41.36$_{\pm 2.85}$	40.32$_{\pm 3.49}$	40.32$_{\pm 5.82}$	41.44$_{\pm 5.02}$	47.90$_{\pm 3.74}$	39.78$_{\pm 3.99}$	67.40$_{\pm 1.51}$	67.44$_{\pm 2.46}$
		200	**73.13**$_{\pm 2.41}$	71.56$_{\pm 2.67}$	54.76$_{\pm 3.46}$	55.00$_{\pm 3.72}$	52.40$_{\pm 3.18}$	58.08$_{\pm 3.52}$	58.48$_{\pm 2.08}$	50.98$_{\pm 2.68}$	70.30$_{\pm 2.91}$	72.38$_{\pm 3.01}$
		500	77.44$_{\pm 1.20}$	76.42$_{\pm 1.28}$	68.28$_{\pm 2.12}$	70.18$_{\pm 1.89}$	68.12$_{\pm 1.62}$	72.36$_{\pm 1.65}$	71.54$_{\pm 1.95}$	69.48$_{\pm 1.71}$	76.52$_{\pm 1.69}$	**77.50**$_{\pm 2.14}$
	biodeg	20	**71.34**$_{\pm 5.63}$	70.10$_{\pm 5.49}$	70.16$_{\pm 5.75}$	58.17$_{\pm 8.00}$	58.05$_{\pm 9.91}$	49.99$_{\pm 5.88}$	62.61$_{\pm 6.45}$	69.47$_{\pm 6.00}$	70.76$_{\pm 3.95}$	71.24$_{\pm 4.85}$
		50	76.35$_{\pm 2.88}$	75.69$_{\pm 3.03}$	73.63$_{\pm 2.64}$	67.44$_{\pm 3.83}$	62.87$_{\pm 7.30}$	49.44$_{\pm 2.63}$	74.44$_{\pm 2.77}$	71.75$_{\pm 5.27}$	75.68$_{\pm 2.31}$	**76.41**$_{\pm 2.93}$
		100	**78.91**$_{\pm 1.40}$	78.39$_{\pm 1.53}$	77.09$_{\pm 2.80}$	74.89$_{\pm 2.54}$	68.62$_{\pm 5.21}$	55.61$_{\pm 3.56}$	75.62$_{\pm 2.77}$	72.45$_{\pm 3.31}$	77.92$_{\pm 2.41}$	78.34$_{\pm 2.18}$
		200	**82.00**$_{\pm 1.47}$	81.42$_{\pm 1.39}$	80.07$_{\pm 1.82}$	78.56$_{\pm 3.43}$	72.35$_{\pm 1.72}$	59.06$_{\pm 4.65}$	78.03$_{\pm 1.95}$	73.73$_{\pm 2.09}$	81.24$_{\pm 1.71}$	81.43$_{\pm 1.78}$
		500	**83.83**$_{\pm 0.57}$	83.74$_{\pm 0.90}$	81.69$_{\pm 0.82}$	82.12$_{\pm 1.17}$	78.06$_{\pm 2.13}$	66.86$_{\pm 5.43}$	81.47$_{\pm 0.93}$	77.98$_{\pm 1.27}$	83.43$_{\pm 0.82}$	83.10$_{\pm 0.98}$
	steel	20	63.66$_{\pm 8.98}$	57.88$_{\pm 5.72}$	60.27$_{\pm 7.47}$	57.90$_{\pm 4.45}$	53.10$_{\pm 7.28}$	54.20$_{\pm 6.99}$	55.41$_{\pm 4.92}$	53.29$_{\pm 4.31}$	66.81$_{\pm 9.74}$	**67.03**$_{\pm 9.35}$
		50	87.91$_{\pm 5.88}$	69.01$_{\pm 6.60}$	66.22$_{\pm 3.63}$	66.22$_{\pm 5.77}$	57.05$_{\pm 5.51}$	57.46$_{\pm 8.48}$	64.81$_{\pm 4.64}$	57.20$_{\pm 5.19}$	**93.63**$_{\pm 4.78}$	92.20$_{\pm 4.81}$
		100	98.85$_{\pm 1.20}$	82.67$_{\pm 4.30}$	74.33$_{\pm 3.85}$	70.49$_{\pm 5.35}$	65.09$_{\pm 7.30}$	52.77$_{\pm 7.06}$	67.85$_{\pm 4.94}$	61.62$_{\pm 4.05}$	**99.24**$_{\pm 0.82}$	99.21$_{\pm 0.86}$
		200	99.43$_{\pm 0.58}$	87.18$_{\pm 3.06}$	82.77$_{\pm 3.21}$	80.34$_{\pm 2.93}$	70.49$_{\pm 5.27}$	72.99$_{\pm 13.98}$	80.27$_{\pm 7.32}$	64.52$_{\pm 2.16}$	99.45$_{\pm 0.69}$	**99.51**$_{\pm 0.69}$
		500	99.75$_{\pm 0.29}$	96.63$_{\pm 2.11}$	94.59$_{\pm 2.98}$	96.32$_{\pm 1.52}$	84.15$_{\pm 2.69}$	98.07$_{\pm 1.37}$	95.35$_{\pm 2.06}$	70.11$_{\pm 2.58}$	99.84$_{\pm 0.20}$	99.84$_{\pm 0.20}$
	stock	20	77.99$_{\pm 4.40}$	80.45$_{\pm 3.98}$	74.21$_{\pm 6.36}$	59.20$_{\pm 12.69}$	72.50$_{\pm 7.92}$	72.09$_{\pm 9.75}$	69.04$_{\pm 6.25}$	**80.59**$_{\pm 3.59}$	79.54$_{\pm 4.46}$	80.39$_{\pm 3.42}$
		50	80.68$_{\pm 2.65}$	81.49$_{\pm 2.95}$	76.41$_{\pm 3.95}$	72.95$_{\pm 2.17}$	75.41$_{\pm 6.00}$	78.44$_{\pm 4.40}$	76.91$_{\pm 2.36}$	75.49$_{\pm 5.31}$	**82.37**$_{\pm 3.20}$	82.21$_{\pm 2.60}$
		100	82.11$_{\pm 1.11}$	**83.86**$_{\pm 1.97}$	79.85$_{\pm 2.79}$	78.47$_{\pm 2.71}$	76.99$_{\pm 3.49}$	80.82$_{\pm 3.57}$	78.89$_{\pm 2.36}$	77.65$_{\pm 2.60}$	83.67$_{\pm 1.60}$	83.52$_{\pm 1.76}$
		200	82.18$_{\pm 0.81}$	**84.29**$_{\pm 1.19}$	79.24$_{\pm 2.82}$	79.86$_{\pm 2.42}$	76.49$_{\pm 1.37}$	80.21$_{\pm 2.13}$	78.87$_{\pm 2.46}$	76.91$_{\pm 1.04}$	83.75$_{\pm 1.53}$	84.17$_{\pm 1.42}$
More than 10 classes	energy	50	**22.22**$_{\pm 2.36}$	N/A	10.11$_{\pm 2.20}$	9.58$_{\pm 3.15}$	7.70$_{\pm 1.83}$	8.20$_{\pm 2.01}$	10.51$_{\pm 1.28}$	17.10$_{\pm 5.03}$	N/A	21.66$_{\pm 1.54}$
		100	24.00$_{\pm 2.30}$	N/A	13.80$_{\pm 2.23}$	13.01$_{\pm 1.71}$	12.14$_{\pm 1.87}$	10.79$_{\pm 3.19}$	15.65$_{\pm 2.40}$	14.45$_{\pm 2.90}$	N/A	**28.10**$_{\pm 2.19}$
		200	29.37$_{\pm 2.63}$	N/A	16.39$_{\pm 2.68}$	16.56$_{\pm 3.58}$	16.78$_{\pm 3.15}$	18.11$_{\pm 1.71}$	20.10$_{\pm 2.48}$	20.92$_{\pm 2.79}$	N/A	**34.38**$_{\pm 2.60}$
	collins	100	**14.28**$_{\pm 1.63}$	N/A	10.57$_{\pm 1.72}$	8.69$_{\pm 1.17}$	9.59$_{\pm 1.35}$	13.31$_{\pm 1.67}$	8.69$_{\pm 1.80}$	12.08$_{\pm 1.56}$	N/A	14.01$_{\pm 2.55}$
		200	19.20$_{\pm 1.71}$	**19.39**$_{\pm 1.88}$	16.03$_{\pm 1.74}$	11.64$_{\pm 1.76}$	10.97$_{\pm 1.46}$	17.06$_{\pm 1.51}$	11.31$_{\pm 1.58}$	17.80$_{\pm 1.21}$	N/A	19.33$_{\pm 1.55}$
	texture	50	86.56$_{\pm 2.96}$	86.93$_{\pm 2.77}$	55.01$_{\pm 5.77}$	42.17$_{\pm 6.36}$	44.63$_{\pm 5.41}$	60.07$_{\pm 10.11}$	44.46$_{\pm 6.63}$	77.68$_{\pm 4.33}$	N/A	**88.54**$_{\pm 2.88}$
		100	94.07$_{\pm 1.70}$	93.87$_{\pm 1.82}$	65.36$_{\pm 4.49}$	60.07$_{\pm 6.81}$	60.76$_{\pm 5.18}$	73.16$_{\pm 5.11}$	64.69$_{\pm 4.79}$	84.13$_{\pm 1.97}$	N/A	**94.38**$_{\pm 1.24}$
		200	**96.65**$_{\pm 1.24}$	96.53$_{\pm 1.33}$	75.91$_{\pm 5.58}$	80.02$_{\pm 5.13}$	77.07$_{\pm 3.89}$	86.24$_{\pm 3.62}$	85.90$_{\pm 2.78}$	85.94$_{\pm 2.88}$	N/A	96.53$_{\pm 1.27}$
		500	98.03$_{\pm 0.36}$	**98.05**$_{\pm 0.23}$	91.87$_{\pm 0.93}$	92.93$_{\pm 1.78}$	90.01$_{\pm 1.80}$	93.92$_{\pm 0.81}$	94.83$_{\pm 0.89}$	91.72$_{\pm 1.49}$	N/A	97.75$_{\pm 0.42}$
Average rank			2.36$_{\pm 1.14}$	3.45$_{\pm 1.35}$	6.52$_{\pm 1.48}$	7.53$_{\pm 1.42}$	9.08$_{\pm 0.77}$	7.61$_{\pm 2.33}$	6.70$_{\pm 1.47}$	6.67$_{\pm 2.53}$	3.17$_{\pm 1.81}$	**1.92**$_{\pm 0.75}$

Table 9: **Classification accuracy** (%) of KNN, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean ± std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable or the downstream predictor failed to converge, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. TabEBM achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	21.34$_{\pm 2.93}$	N/A	21.78$_{\pm 2.06}$	21.18$_{\pm 4.22}$	21.30$_{\pm 1.90}$	22.00$_{\pm 2.70}$	22.69$_{\pm 3.86}$	16.99$_{\pm 3.45}$	**35.78**$_{\pm 4.46}$	35.76$_{\pm 4.37}$
		50	36.41$_{\pm 4.33}$	**55.24**$_{\pm 3.81}$	35.85$_{\pm 2.50}$	36.13$_{\pm 4.24}$	35.40$_{\pm 4.27}$	36.77$_{\pm 4.06}$	36.84$_{\pm 4.05}$	31.02$_{\pm 4.11}$	53.38$_{\pm 3.53}$	53.49$_{\pm 3.30}$
		100	50.17$_{\pm 3.11}$	**70.11**$_{\pm 2.82}$	51.97$_{\pm 2.84}$	50.61$_{\pm 3.15}$	50.62$_{\pm 3.27}$	50.63$_{\pm 3.55}$	50.36$_{\pm 3.44}$	44.70$_{\pm 2.22}$	67.99$_{\pm 2.43}$	68.27$_{\pm 2.51}$
		200	65.84$_{\pm 2.78}$	80.43$_{\pm 2.44}$	65.52$_{\pm 2.96}$	66.05$_{\pm 2.74}$	66.14$_{\pm 2.42}$	67.50$_{\pm 2.57}$	66.52$_{\pm 3.16}$	63.92$_{\pm 3.26}$	79.94$_{\pm 2.26}$	**80.55**$_{\pm 2.02}$
		500	85.63$_{\pm 1.41}$	90.92$_{\pm 1.42}$	87.08$_{\pm 1.86}$	85.77$_{\pm 1.43}$	85.51$_{\pm 1.50}$	86.47$_{\pm 1.40}$	85.87$_{\pm 1.63}$	85.64$_{\pm 1.94}$	91.32$_{\pm 1.09}$	**91.67**$_{\pm 1.11}$
	fourier	20	18.06$_{\pm 3.30}$	N/A	26.56$_{\pm 4.92}$	24.88$_{\pm 3.66}$	19.80$_{\pm 3.77}$	19.30$_{\pm 3.53}$	23.42$_{\pm 3.45}$	18.78$_{\pm 2.17}$	41.08$_{\pm 6.56}$	**42.78**$_{\pm 5.83}$
		50	48.00$_{\pm 2.47}$	**60.38**$_{\pm 1.67}$	39.86$_{\pm 3.73}$	46.82$_{\pm 3.52}$	43.56$_{\pm 3.45}$	49.54$_{\pm 2.78}$	42.98$_{\pm 2.80}$	28.12$_{\pm 2.75}$	59.50$_{\pm 1.99}$	58.54$_{\pm 1.86}$
		100	58.36$_{\pm 3.26}$	**66.96**$_{\pm 2.47}$	48.44$_{\pm 4.14}$	53.94$_{\pm 3.47}$	53.50$_{\pm 2.54}$	60.80$_{\pm 4.28}$	52.74$_{\pm 3.15}$	35.70$_{\pm 2.40}$	63.88$_{\pm 2.53}$	65.08$_{\pm 2.47}$
		200	68.60$_{\pm 2.55}$	**71.90**$_{\pm 2.02}$	59.66$_{\pm 3.31}$	66.54$_{\pm 2.75}$	65.16$_{\pm 2.71}$	70.22$_{\pm 2.26}$	64.52$_{\pm 2.44}$	51.24$_{\pm 7.29}$	70.32$_{\pm 1.94}$	71.08$_{\pm 1.87}$
		500	76.90$_{\pm 1.30}$	77.64$_{\pm 1.07}$	73.20$_{\pm 1.68}$	76.22$_{\pm 1.62}$	75.72$_{\pm 1.40}$	**78.88**$_{\pm 1.58}$	76.54$_{\pm 0.77}$	63.66$_{\pm 2.49}$	74.30$_{\pm 1.51}$	75.35$_{\pm 1.34}$
	biodeg	20	65.23$_{\pm 5.01}$	68.99$_{\pm 3.31}$	66.63$_{\pm 7.83}$	56.99$_{\pm 5.55}$	59.91$_{\pm 6.09}$	55.85$_{\pm 4.94}$	58.77$_{\pm 5.93}$	56.62$_{\pm 7.29}$	67.79$_{\pm 4.64}$	**69.76**$_{\pm 4.43}$
		50	71.26$_{\pm 3.13}$	73.19$_{\pm 2.46}$	70.80$_{\pm 2.14}$	70.00$_{\pm 5.92}$	65.90$_{\pm 3.57}$	73.50$_{\pm 4.43}$	70.23$_{\pm 3.35}$	65.29$_{\pm 4.57}$	72.08$_{\pm 3.84}$	**73.58**$_{\pm 3.57}$
		100	76.12$_{\pm 1.98}$	76.07$_{\pm 1.74}$	74.02$_{\pm 2.78}$	75.36$_{\pm 2.18}$	73.24$_{\pm 2.61}$	**77.34**$_{\pm 2.19}$	74.28$_{\pm 2.02}$	72.26$_{\pm 2.46}$	74.56$_{\pm 1.58}$	75.60$_{\pm 1.55}$
		200	78.86$_{\pm 2.19}$	**79.67**$_{\pm 1.68}$	77.31$_{\pm 2.93}$	78.05$_{\pm 3.07}$	77.64$_{\pm 2.71}$	77.84$_{\pm 2.62}$	78.81$_{\pm 2.66}$	76.82$_{\pm 2.29}$	77.46$_{\pm 1.68}$	78.46$_{\pm 1.69}$
		500	82.59$_{\pm 1.17}$	**83.07**$_{\pm 1.50}$	82.13$_{\pm 1.21}$	82.17$_{\pm 1.32}$	82.80$_{\pm 1.28}$	81.06$_{\pm 1.22}$	82.15$_{\pm 1.33}$	82.10$_{\pm 0.79}$	79.99$_{\pm 1.76}$	81.01$_{\pm 1.66}$
	steel	20	56.40$_{\pm 4.48}$	63.95$_{\pm 3.14}$	59.45$_{\pm 8.27}$	57.04$_{\pm 5.05}$	54.59$_{\pm 5.81}$	65.46$_{\pm 6.10}$	56.97$_{\pm 5.43}$	52.90$_{\pm 3.76}$	**70.68**$_{\pm 3.87}$	69.31$_{\pm 4.02}$
		50	73.95$_{\pm 4.76}$	70.24$_{\pm 3.44}$	67.60$_{\pm 4.10}$	68.77$_{\pm 2.85}$	67.00$_{\pm 4.58}$	**85.14**$_{\pm 8.76}$	64.02$_{\pm 3.71}$	57.54$_{\pm 2.34}$	82.09$_{\pm 3.09}$	80.47$_{\pm 3.48}$
		100	84.70$_{\pm 5.57}$	77.46$_{\pm 3.67}$	71.87$_{\pm 2.98}$	72.94$_{\pm 4.62}$	77.09$_{\pm 2.63}$	**94.05**$_{\pm 3.84}$	72.62$_{\pm 5.21}$	61.08$_{\pm 1.93}$	87.77$_{\pm 3.13}$	87.67$_{\pm 3.22}$
		200	90.44$_{\pm 2.80}$	82.46$_{\pm 1.43}$	80.83$_{\pm 2.65}$	82.73$_{\pm 3.88}$	85.49$_{\pm 3.59}$	**98.99**$_{\pm 0.75}$	83.38$_{\pm 2.67}$	69.12$_{\pm 2.56}$	92.01$_{\pm 1.73}$	92.06$_{\pm 1.48}$
		500	94.99$_{\pm 1.09}$	89.97$_{\pm 0.88}$	91.34$_{\pm 1.69}$	92.42$_{\pm 1.36}$	93.37$_{\pm 1.13}$	**99.71**$_{\pm 0.21}$	92.02$_{\pm 2.03}$	80.79$_{\pm 1.93}$	95.08$_{\pm 1.30}$	95.50$_{\pm 1.49}$
	stock	20	71.89$_{\pm 4.37}$	84.41$_{\pm 5.28}$	73.80$_{\pm 4.68}$	66.38$_{\pm 9.10}$	68.93$_{\pm 10.49}$	81.82$_{\pm 8.38}$	67.53$_{\pm 8.58}$	71.80$_{\pm 4.99}$	84.41$_{\pm 4.22}$	**84.69**$_{\pm 4.16}$
		50	85.03$_{\pm 3.39}$	**89.77**$_{\pm 1.99}$	84.32$_{\pm 3.97}$	83.49$_{\pm 3.67}$	84.43$_{\pm 2.04}$	89.34$_{\pm 1.59}$	84.33$_{\pm 3.22}$	83.64$_{\pm 2.53}$	89.67$_{\pm 1.88}$	89.68$_{\pm 1.87}$
		100	89.66$_{\pm 1.39}$	92.32$_{\pm 0.99}$	89.58$_{\pm 1.22}$	89.61$_{\pm 1.36}$	89.66$_{\pm 1.01}$	91.40$_{\pm 1.41}$	89.66$_{\pm 2.10}$	89.44$_{\pm 1.41}$	92.02$_{\pm 0.81}$	**92.47**$_{\pm 0.83}$
		200	91.65$_{\pm 1.08}$	93.46$_{\pm 0.82}$	92.37$_{\pm 1.18}$	91.55$_{\pm 1.19}$	91.43$_{\pm 1.34}$	92.92$_{\pm 1.00}$	91.14$_{\pm 1.58}$	91.53$_{\pm 1.05}$	93.15$_{\pm 0.72}$	**93.62**$_{\pm 1.14}$
More than 10 classes	energy	50	10.85$_{\pm 1.76}$	N/A	10.64$_{\pm 2.36}$	8.22$_{\pm 2.03}$	8.83$_{\pm 1.53}$	8.92$_{\pm 2.51}$	9.14$_{\pm 1.95}$	11.86$_{\pm 2.33}$	N/A	**25.36**$_{\pm 2.27}$
		100	18.60$_{\pm 1.83}$	N/A	13.71$_{\pm 1.66}$	15.81$_{\pm 1.50}$	14.67$_{\pm 1.55}$	16.18$_{\pm 1.75}$	15.71$_{\pm 2.79}$	17.64$_{\pm 2.68}$	N/A	**29.82**$_{\pm 2.74}$
		200	26.45$_{\pm 1.49}$	N/A	20.71$_{\pm 1.02}$	21.71$_{\pm 3.23}$	23.40$_{\pm 2.15}$	23.95$_{\pm 2.94}$	23.09$_{\pm 2.56}$	27.35$_{\pm 2.28}$	N/A	**35.93**$_{\pm 2.85}$
	collins	100	10.59$_{\pm 1.48}$	N/A	7.58$_{\pm 0.74}$	7.95$_{\pm 1.12}$	7.55$_{\pm 1.32}$	14.24$_{\pm 1.48}$	7.42$_{\pm 1.17}$	8.79$_{\pm 0.93}$	N/A	**15.16**$_{\pm 1.92}$
		200	15.84$_{\pm 1.74}$	**19.81**$_{\pm 1.73}$	9.79$_{\pm 1.14}$	11.21$_{\pm 1.45}$	12.24$_{\pm 1.65}$	16.30$_{\pm 1.54}$	10.96$_{\pm 1.43}$	12.86$_{\pm 1.50}$	N/A	18.05$_{\pm 1.65}$
	texture	50	62.96$_{\pm 2.49}$	**78.80**$_{\pm 2.75}$	55.51$_{\pm 3.69}$	61.86$_{\pm 4.48}$	62.08$_{\pm 3.17}$	61.91$_{\pm 2.24}$	62.67$_{\pm 2.29}$	56.81$_{\pm 2.98}$	N/A	75.57$_{\pm 2.67}$
		100	77.16$_{\pm 1.25}$	**86.15**$_{\pm 2.62}$	69.54$_{\pm 2.66}$	76.53$_{\pm 2.22}$	76.85$_{\pm 1.56}$	77.77$_{\pm 1.80}$	76.70$_{\pm 2.05}$	72.64$_{\pm 1.81}$	N/A	84.83$_{\pm 1.67}$
		200	85.34$_{\pm 1.18}$	89.07$_{\pm 1.74}$	81.70$_{\pm 1.32}$	85.46$_{\pm 1.28}$	84.62$_{\pm 1.09}$	85.94$_{\pm 1.35}$	85.11$_{\pm 1.20}$	84.72$_{\pm 0.80}$	N/A	**89.48**$_{\pm 2.01}$
		500	91.40$_{\pm 1.60}$	93.14$_{\pm 1.28}$	89.88$_{\pm 1.44}$	91.40$_{\pm 1.55}$	91.34$_{\pm 1.60}$	92.31$_{\pm 1.60}$	91.46$_{\pm 1.51}$	91.91$_{\pm 1.63}$	N/A	**93.46**$_{\pm 0.66}$
Average rank			5.15$_{\pm 2.06}$	2.70$_{\pm 1.97}$	7.67$_{\pm 2.10}$	7.03$_{\pm 1.55}$	7.27$_{\pm 1.68}$	4.12$_{\pm 2.34}$	6.82$_{\pm 1.76}$	8.42$_{\pm 2.33}$	3.67$_{\pm 1.96}$	**2.15**$_{\pm 1.75}$

Table 10: **Classification accuracy** (%) of MLP, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean ± std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable or the downstream predictor failed to converge, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. TabEBM achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	35.12$_{\pm 2.59}$	N/A	21.89$_{\pm 3.62}$	26.95$_{\pm 4.13}$	24.56$_{\pm 5.06}$	27.30$_{\pm 3.23}$	27.96$_{\pm 4.51}$	27.09$_{\pm 3.47}$	36.19$_{\pm 2.84}$	**36.26**$_{\pm 2.65}$
		50	58.11$_{\pm 4.13}$	57.24$_{\pm 4.60}$	40.77$_{\pm 3.59}$	43.04$_{\pm 5.43}$	44.78$_{\pm 3.92}$	49.43$_{\pm 2.51}$	46.84$_{\pm 5.08}$	44.78$_{\pm 2.67}$	58.62$_{\pm 4.41}$	**58.75**$_{\pm 4.48}$
		100	76.82$_{\pm 3.33}$	76.78$_{\pm 3.10}$	62.00$_{\pm 3.21}$	64.14$_{\pm 3.19}$	63.24$_{\pm 4.05}$	69.20$_{\pm 2.75}$	65.08$_{\pm 2.67}$	62.45$_{\pm 3.22}$	**77.84**$_{\pm 3.49}$	77.63$_{\pm 3.69}$
		200	89.53$_{\pm 2.34}$	90.28$_{\pm 2.13}$	80.28$_{\pm 3.49}$	81.94$_{\pm 2.56}$	81.85$_{\pm 2.46}$	85.48$_{\pm 2.07}$	82.57$_{\pm 2.19}$	78.04$_{\pm 2.66}$	**90.74**$_{\pm 2.13}$	90.48$_{\pm 2.06}$
		500	98.23$_{\pm 0.91}$	98.25$_{\pm 0.78}$	95.08$_{\pm 1.34}$	95.74$_{\pm 0.95}$	96.15$_{\pm 1.09}$	96.01$_{\pm 0.86}$	96.50$_{\pm 0.87}$	96.23$_{\pm 1.67}$	**98.52**$_{\pm 0.80}$	98.50$_{\pm 0.70}$
	fourier	20	33.66$_{\pm 3.92}$	N/A	23.20$_{\pm 5.54}$	17.08$_{\pm 2.75}$	19.40$_{\pm 4.03}$	18.32$_{\pm 3.82}$	23.26$_{\pm 4.21}$	19.64$_{\pm 2.40}$	**37.00**$_{\pm 2.85}$	35.02$_{\pm 3.77}$
		50	53.72$_{\pm 1.67}$	53.02$_{\pm 1.96}$	37.16$_{\pm 3.08}$	37.60$_{\pm 4.52}$	35.14$_{\pm 2.44}$	40.90$_{\pm 2.80}$	42.82$_{\pm 2.83}$	32.66$_{\pm 5.19}$	**55.40**$_{\pm 2.23}$	55.34$_{\pm 1.40}$
		100	62.78$_{\pm 1.60}$	61.44$_{\pm 2.74}$	43.68$_{\pm 3.15}$	48.80$_{\pm 2.66}$	46.18$_{\pm 3.96}$	56.52$_{\pm 5.04}$	52.50$_{\pm 2.66}$	37.74$_{\pm 2.99}$	63.00$_{\pm 1.95}$	**63.54**$_{\pm 1.83}$
		200	70.18$_{\pm 1.85}$	70.06$_{\pm 2.10}$	58.90$_{\pm 2.56}$	62.36$_{\pm 2.86}$	58.40$_{\pm 2.53}$	70.08$_{\pm 1.90}$	62.14$_{\pm 2.00}$	50.92$_{\pm 5.13}$	**71.49**$_{\pm 1.41}$	71.36$_{\pm 1.36}$
		500	77.94$_{\pm 1.65}$	77.18$_{\pm 1.35}$	72.14$_{\pm 1.79}$	74.30$_{\pm 1.65}$	71.38$_{\pm 1.54}$	77.78$_{\pm 1.26}$	74.32$_{\pm 1.56}$	67.28$_{\pm 2.91}$	78.34$_{\pm 1.72}$	**79.30**$_{\pm 0.99}$
	biodeg	20	71.31$_{\pm 5.13}$	68.84$_{\pm 5.95}$	66.64$_{\pm 8.27}$	62.11$_{\pm 4.95}$	62.61$_{\pm 6.78}$	52.96$_{\pm 4.22}$	62.06$_{\pm 3.69}$	65.81$_{\pm 7.24}$	72.04$_{\pm 5.12}$	**72.09**$_{\pm 4.81}$
		50	76.73$_{\pm 3.16}$	74.97$_{\pm 2.51}$	72.02$_{\pm 4.74}$	71.83$_{\pm 3.17}$	67.86$_{\pm 6.02}$	69.92$_{\pm 4.83}$	74.03$_{\pm 3.05}$	71.01$_{\pm 2.78}$	**77.17**$_{\pm 2.93}$	77.11$_{\pm 3.20}$
		100	**79.13**$_{\pm 1.91}$	78.20$_{\pm 1.68}$	76.78$_{\pm 2.79}$	77.85$_{\pm 2.73}$	76.01$_{\pm 2.88}$	76.74$_{\pm 3.62}$	76.08$_{\pm 2.39}$	76.24$_{\pm 2.45}$	78.23$_{\pm 2.29}$	79.08$_{\pm 2.03}$
		200	**82.39**$_{\pm 1.48}$	81.70$_{\pm 1.22}$	80.43$_{\pm 2.02}$	79.96$_{\pm 2.35}$	79.92$_{\pm 1.55}$	80.51$_{\pm 1.26}$	79.59$_{\pm 1.72}$	80.34$_{\pm 1.93}$	81.74$_{\pm 1.36}$	82.24$_{\pm 1.54}$
		500	**84.50**$_{\pm 0.61}$	84.50$_{\pm 0.81}$	83.78$_{\pm 1.51}$	83.67$_{\pm 0.81}$	84.13$_{\pm 1.20}$	84.09$_{\pm 0.84}$	83.76$_{\pm 1.27}$	82.97$_{\pm 1.21}$	84.37$_{\pm 0.48}$	84.14$_{\pm 0.59}$
	steel	20	62.35$_{\pm 6.30}$	60.34$_{\pm 5.73}$	61.63$_{\pm 8.82}$	59.09$_{\pm 4.25}$	56.99$_{\pm 7.13}$	60.67$_{\pm 9.18}$	55.23$_{\pm 3.92}$	55.78$_{\pm 3.01}$	**64.49**$_{\pm 6.03}$	64.22$_{\pm 5.89}$
		50	79.65$_{\pm 5.53}$	68.18$_{\pm 3.16}$	69.01$_{\pm 3.48}$	70.30$_{\pm 4.77}$	66.96$_{\pm 5.12}$	**84.04**$_{\pm 8.03}$	64.79$_{\pm 4.07}$	58.95$_{\pm 1.66}$	82.72$_{\pm 6.02}$	82.15$_{\pm 5.78}$
		100	92.18$_{\pm 2.93}$	78.44$_{\pm 3.67}$	76.37$_{\pm 3.06}$	76.92$_{\pm 3.63}$	76.50$_{\pm 3.18}$	**95.83**$_{\pm 1.90}$	71.85$_{\pm 3.20}$	67.35$_{\pm 2.51}$	95.16$_{\pm 3.13}$	95.41$_{\pm 3.23}$
		200	97.31$_{\pm 1.63}$	83.93$_{\pm 1.83}$	82.42$_{\pm 2.75}$	84.70$_{\pm 2.54}$	84.06$_{\pm 4.46}$	98.75$_{\pm 0.68}$	79.66$_{\pm 3.26}$	78.36$_{\pm 3.86}$	98.83$_{\pm 0.80}$	**98.84**$_{\pm 0.67}$
		500	99.78$_{\pm 0.30}$	91.37$_{\pm 2.26}$	93.08$_{\pm 1.82}$	94.82$_{\pm 1.54}$	93.99$_{\pm 1.83}$	99.47$_{\pm 0.44}$	90.34$_{\pm 2.19}$	96.06$_{\pm 1.03}$	**99.81**$_{\pm 0.24}$	99.81$_{\pm 0.24}$
	stock	20	83.56$_{\pm 3.89}$	83.62$_{\pm 4.06}$	77.25$_{\pm 5.02}$	69.90$_{\pm 9.27}$	72.85$_{\pm 7.77}$	80.60$_{\pm 5.73}$	69.30$_{\pm 6.76}$	83.34$_{\pm 3.38}$	83.81$_{\pm 3.92}$	**83.89**$_{\pm 4.05}$
		50	89.57$_{\pm 2.01}$	89.71$_{\pm 2.21}$	82.62$_{\pm 3.49}$	79.35$_{\pm 2.71}$	81.52$_{\pm 1.99}$	88.48$_{\pm 2.18}$	83.36$_{\pm 2.30}$	88.38$_{\pm 2.41}$	90.23$_{\pm 2.02}$	**90.38**$_{\pm 2.17}$
		100	90.63$_{\pm 0.83}$	91.17$_{\pm 0.83}$	88.37$_{\pm 2.40}$	86.60$_{\pm 3.27}$	83.19$_{\pm 3.77}$	90.65$_{\pm 1.39}$	88.64$_{\pm 1.15}$	91.61$_{\pm 0.70}$	91.70$_{\pm 1.17}$	**91.75**$_{\pm 0.97}$
		200	91.25$_{\pm 0.74}$	92.58$_{\pm 0.91}$	91.27$_{\pm 0.95}$	90.34$_{\pm 1.60}$	89.32$_{\pm 2.45}$	92.19$_{\pm 0.78}$	90.37$_{\pm 1.32}$	91.89$_{\pm 1.32}$	**92.91**$_{\pm 0.62}$	92.47$_{\pm 0.63}$
More than 10 classes	energy	50	**24.79**$_{\pm 1.76}$	N/A	12.51$_{\pm 2.89}$	12.61$_{\pm 3.45}$	8.43$_{\pm 2.11}$	10.91$_{\pm 2.11}$	12.45$_{\pm 1.87}$	20.42$_{\pm 4.41}$	N/A	24.04$_{\pm 1.39}$
		100	26.86$_{\pm 1.51}$	N/A	16.20$_{\pm 2.12}$	15.78$_{\pm 2.50}$	15.70$_{\pm 2.44}$	17.75$_{\pm 3.18}$	18.16$_{\pm 2.46}$	20.72$_{\pm 2.99}$	N/A	**29.30**$_{\pm 2.32}$
		200	33.36$_{\pm 2.98}$	N/A	22.30$_{\pm 2.44}$	23.00$_{\pm 4.24}$	23.53$_{\pm 1.84}$	26.12$_{\pm 1.78}$	26.28$_{\pm 3.03}$	33.03$_{\pm 3.38}$	N/A	**41.27**$_{\pm 2.93}$
	collins	100	**14.16**$_{\pm 1.31}$	N/A	9.24$_{\pm 1.71}$	9.16$_{\pm 1.57}$	9.04$_{\pm 1.79}$	14.03$_{\pm 1.24}$	8.59$_{\pm 1.84}$	10.81$_{\pm 1.68}$	N/A	14.07$_{\pm 1.58}$
		200	19.35$_{\pm 1.24}$	19.06$_{\pm 1.49}$	14.62$_{\pm 2.00}$	13.17$_{\pm 0.94}$	12.38$_{\pm 1.56}$	18.63$_{\pm 1.56}$	12.48$_{\pm 1.91}$	17.65$_{\pm 1.76}$	N/A	**19.53**$_{\pm 1.44}$
	texture	50	84.50$_{\pm 2.81}$	84.12$_{\pm 3.02}$	62.92$_{\pm 4.09}$	67.69$_{\pm 5.49}$	61.99$_{\pm 3.58}$	69.69$_{\pm 4.62}$	64.45$_{\pm 7.84}$	69.68$_{\pm 3.53}$	N/A	**85.51**$_{\pm 2.89}$
		100	91.50$_{\pm 1.34}$	91.57$_{\pm 1.59}$	74.53$_{\pm 3.39}$	79.96$_{\pm 4.37}$	80.36$_{\pm 4.58}$	85.42$_{\pm 2.74}$	80.23$_{\pm 2.18}$	85.59$_{\pm 1.49}$	N/A	**92.17**$_{\pm 1.31}$
		200	93.81$_{\pm 1.35}$	94.18$_{\pm 1.26}$	86.57$_{\pm 2.33}$	90.68$_{\pm 1.55}$	88.97$_{\pm 2.12}$	90.10$_{\pm 2.26}$	89.14$_{\pm 1.85}$	91.66$_{\pm 1.43}$	N/A	**94.35**$_{\pm 1.57}$
		500	96.55$_{\pm 0.63}$	**97.21**$_{\pm 0.40}$	94.66$_{\pm 1.17}$	96.27$_{\pm 0.74}$	94.34$_{\pm 1.36}$	94.72$_{\pm 0.61}$	95.83$_{\pm 1.03}$	96.49$_{\pm 0.48}$	N/A	97.13$_{\pm 0.53}$
Average rank			3.00$_{\pm 1.32}$	4.06$_{\pm 1.62}$	7.82$_{\pm 1.63}$	7.48$_{\pm 1.50}$	8.45$_{\pm 1.33}$	5.55$_{\pm 2.14}$	7.48$_{\pm 1.72}$	6.82$_{\pm 2.57}$	2.67$_{\pm 1.69}$	**1.67**$_{\pm 0.74}$

Table 11: **Classification accuracy** (%) of RF, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean ± std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable or the downstream predictor failed to converge, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. TabEBM achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	$28.52_{\pm 2.19}$	N/A	$22.74_{\pm 4.23}$	$24.94_{\pm 5.15}$	$24.61_{\pm 2.46}$	$29.62_{\pm 4.06}$	$27.69_{\pm 3.73}$	$25.76_{\pm 1.62}$	$32.04_{\pm 2.40}$	**34.19**$_{\pm 2.21}$
		50	$53.40_{\pm 3.26}$	$55.69_{\pm 2.61}$	$46.95_{\pm 3.13}$	$43.91_{\pm 4.98}$	$43.28_{\pm 4.07}$	$47.93_{\pm 4.49}$	$44.48_{\pm 3.35}$	$47.25_{\pm 4.81}$	$54.29_{\pm 2.57}$	**56.85**$_{\pm 2.49}$
		100	$68.13_{\pm 3.19}$	**72.89**$_{\pm 2.60}$	$63.24_{\pm 1.78}$	$61.19_{\pm 2.10}$	$59.64_{\pm 3.48}$	$65.19_{\pm 3.22}$	$60.05_{\pm 2.88}$	$65.05_{\pm 3.00}$	$71.47_{\pm 3.61}$	$72.57_{\pm 2.50}$
		200	$80.34_{\pm 2.35}$	$83.60_{\pm 2.71}$	$78.51_{\pm 2.58}$	$75.84_{\pm 1.61}$	$76.84_{\pm 2.35}$	$78.74_{\pm 2.24}$	$75.61_{\pm 2.90}$	$79.44_{\pm 2.73}$	$83.36_{\pm 2.40}$	**84.30**$_{\pm 1.97}$
		500	$93.01_{\pm 1.12}$	$93.82_{\pm 0.67}$	$92.86_{\pm 1.66}$	$91.37_{\pm 1.25}$	$93.00_{\pm 1.08}$	$92.93_{\pm 0.97}$	$92.38_{\pm 0.90}$	$92.95_{\pm 0.92}$	**94.49**$_{\pm 1.16}$	$93.94_{\pm 1.23}$
	fourier	20	$35.10_{\pm 4.56}$	N/A	$19.06_{\pm 3.91}$	$17.52_{\pm 2.84}$	$20.78_{\pm 2.54}$	$16.98_{\pm 2.31}$	$23.78_{\pm 3.12}$	$19.00_{\pm 2.93}$	$34.88_{\pm 5.93}$	**38.60**$_{\pm 5.66}$
		50	$64.10_{\pm 3.80}$	$64.76_{\pm 4.00}$	$37.20_{\pm 3.35}$	$32.82_{\pm 4.56}$	$37.78_{\pm 3.11}$	$51.76_{\pm 3.50}$	$47.22_{\pm 4.35}$	$53.86_{\pm 3.61}$	**66.92**$_{\pm 3.05}$	$66.26_{\pm 3.16}$
		100	$73.86_{\pm 3.06}$	$73.78_{\pm 3.22}$	$64.40_{\pm 2.51}$	$60.82_{\pm 3.71}$	$51.64_{\pm 4.16}$	$66.14_{\pm 1.91}$	$58.62_{\pm 3.73}$	$68.16_{\pm 3.12}$	$73.13_{\pm 2.70}$	**74.84**$_{\pm 3.10}$
		200	$78.54_{\pm 2.15}$	$79.18_{\pm 1.92}$	$74.86_{\pm 1.60}$	$74.26_{\pm 2.20}$	$69.36_{\pm 2.61}$	$76.42_{\pm 1.95}$	$72.88_{\pm 1.22}$	$76.64_{\pm 1.99}$	**82.20**$_{\pm 0.85}$	$79.18_{\pm 2.08}$
		500	$81.84_{\pm 1.01}$	$82.14_{\pm 1.49}$	$81.02_{\pm 1.59}$	$81.18_{\pm 1.43}$	$80.08_{\pm 1.62}$	$81.26_{\pm 1.40}$	$80.28_{\pm 1.54}$	$80.62_{\pm 1.52}$	$81.45_{\pm 1.45}$	**83.40**$_{\pm 1.24}$
	biodeg	20	$61.11_{\pm 7.87}$	**68.38**$_{\pm 5.90}$	$65.44_{\pm 8.89}$	$56.29_{\pm 7.96}$	$58.19_{\pm 6.60}$	$52.90_{\pm 4.74}$	$62.33_{\pm 6.14}$	$63.52_{\pm 7.29}$	$67.15_{\pm 5.74}$	$67.82_{\pm 5.13}$
		50	$68.38_{\pm 4.82}$	$70.64_{\pm 3.44}$	$71.77_{\pm 2.99}$	$66.78_{\pm 4.89}$	$61.39_{\pm 4.94}$	$63.98_{\pm 3.65}$	$68.78_{\pm 5.22}$	$70.34_{\pm 3.39}$	$71.38_{\pm 3.60}$	**72.12**$_{\pm 3.29}$
		100	$73.19_{\pm 2.46}$	$75.36_{\pm 2.56}$	$74.98_{\pm 2.58}$	$72.68_{\pm 2.98}$	$69.62_{\pm 3.53}$	$73.11_{\pm 2.39}$	$72.16_{\pm 2.58}$	$74.22_{\pm 2.32}$	**75.85**$_{\pm 1.56}$	$75.65_{\pm 1.53}$
		200	$77.85_{\pm 2.72}$	$78.86_{\pm 1.97}$	$76.42_{\pm 2.25}$	$76.68_{\pm 2.77}$	$73.43_{\pm 3.01}$	$76.16_{\pm 2.00}$	$75.79_{\pm 2.49}$	$77.42_{\pm 2.24}$	**79.68**$_{\pm 1.74}$	$79.22_{\pm 1.70}$
		500	$81.42_{\pm 0.73}$	$82.03_{\pm 1.02}$	$81.88_{\pm 0.87}$	$81.71_{\pm 1.54}$	$80.50_{\pm 1.21}$	$81.43_{\pm 1.26}$	$81.34_{\pm 1.58}$	$81.94_{\pm 0.85}$	**82.38**$_{\pm 1.35}$	$82.10_{\pm 1.31}$
	steel	20	$52.77_{\pm 1.60}$	$56.16_{\pm 4.50}$	$57.23_{\pm 3.97}$	$54.65_{\pm 3.40}$	$53.75_{\pm 3.49}$	$51.70_{\pm 1.66}$	$54.09_{\pm 4.36}$	$55.50_{\pm 2.97}$	$57.04_{\pm 3.07}$	**57.41**$_{\pm 2.67}$
		50	$59.75_{\pm 3.11}$	$62.12_{\pm 2.46}$	$60.65_{\pm 1.96}$	$58.09_{\pm 1.75}$	$54.69_{\pm 2.44}$	$58.14_{\pm 4.21}$	$57.67_{\pm 2.52}$	$60.34_{\pm 2.90}$	$65.07_{\pm 3.11}$	**67.74**$_{\pm 3.36}$
		100	$64.97_{\pm 2.05}$	$69.08_{\pm 3.62}$	$64.46_{\pm 4.17}$	$61.62_{\pm 1.98}$	$58.43_{\pm 2.46}$	$60.53_{\pm 3.64}$	$62.71_{\pm 3.43}$	$63.07_{\pm 2.23}$	$73.28_{\pm 3.39}$	**79.63**$_{\pm 3.41}$
		200	$75.45_{\pm 3.26}$	$74.71_{\pm 3.79}$	$71.45_{\pm 2.18}$	$68.52_{\pm 3.80}$	$62.15_{\pm 2.80}$	$68.10_{\pm 3.59}$	$67.61_{\pm 1.81}$	$67.36_{\pm 1.63}$	$85.12_{\pm 4.44}$	**88.85**$_{\pm 5.10}$
		500	$90.93_{\pm 2.83}$	$85.37_{\pm 2.36}$	$85.63_{\pm 3.14}$	$84.51_{\pm 3.22}$	$76.12_{\pm 2.70}$	$89.19_{\pm 3.20}$	$81.44_{\pm 2.64}$	$80.35_{\pm 3.54}$	$94.35_{\pm 1.34}$	**95.90**$_{\pm 1.06}$
	stock	20	$79.47_{\pm 5.83}$	$81.99_{\pm 4.49}$	$77.94_{\pm 5.21}$	$72.53_{\pm 7.16}$	$73.20_{\pm 9.98}$	$80.99_{\pm 7.01}$	$72.57_{\pm 8.76}$	$78.10_{\pm 5.91}$	$83.96_{\pm 5.57}$	**84.73**$_{\pm 3.46}$
		50	$87.57_{\pm 2.60}$	$89.69_{\pm 1.99}$	$86.62_{\pm 3.44}$	$83.75_{\pm 4.32}$	$84.28_{\pm 2.98}$	$88.69_{\pm 2.11}$	$84.92_{\pm 2.09}$	$88.65_{\pm 2.55}$	$89.35_{\pm 2.18}$	**89.99**$_{\pm 2.63}$
		100	$91.44_{\pm 1.59}$	$91.47_{\pm 2.16}$	$91.07_{\pm 2.08}$	$89.82_{\pm 2.69}$	$89.33_{\pm 1.92}$	$91.33_{\pm 2.07}$	$90.48_{\pm 2.38}$	$92.00_{\pm 2.36}$	$92.07_{\pm 1.22}$	**92.17**$_{\pm 1.24}$
		200	$93.52_{\pm 0.80}$	**93.94**$_{\pm 1.09}$	$93.35_{\pm 1.05}$	$92.62_{\pm 1.02}$	$92.77_{\pm 1.25}$	$93.65_{\pm 1.08}$	$93.08_{\pm 0.53}$	$93.87_{\pm 1.25}$	$93.65_{\pm 1.02}$	$93.67_{\pm 1.07}$
More than 10 classes	energy	50	$18.96_{\pm 1.40}$	N/A	$16.63_{\pm 2.27}$	$15.66_{\pm 2.43}$	$14.81_{\pm 3.16}$	$14.49_{\pm 1.26}$	$15.05_{\pm 3.06}$	$15.58_{\pm 3.26}$	N/A	**27.74**$_{\pm 3.71}$
		100	$30.85_{\pm 2.19}$	N/A	$24.59_{\pm 2.27}$	$28.59_{\pm 2.63}$	$27.59_{\pm 2.86}$	$27.23_{\pm 2.39}$	$27.99_{\pm 2.18}$	$25.43_{\pm 2.46}$	N/A	**41.03**$_{\pm 2.24}$
		200	$45.80_{\pm 2.32}$	N/A	$42.10_{\pm 2.57}$	$41.69_{\pm 3.84}$	$44.41_{\pm 2.51}$	$44.58_{\pm 1.37}$	$41.33_{\pm 3.90}$	$44.64_{\pm 2.54}$	N/A	**53.87**$_{\pm 2.81}$
	collins	100	$10.41_{\pm 1.61}$	N/A	$6.75_{\pm 0.69}$	$8.23_{\pm 1.76}$	$7.34_{\pm 1.46}$	$12.84_{\pm 1.61}$	$6.73_{\pm 1.36}$	$8.43_{\pm 0.94}$	N/A	**13.35**$_{\pm 1.49}$
		200	$13.75_{\pm 1.12}$	**17.56**$_{\pm 1.79}$	$10.51_{\pm 1.41}$	$11.00_{\pm 1.37}$	$9.85_{\pm 1.38}$	$15.15_{\pm 1.22}$	$9.90_{\pm 0.72}$	$13.40_{\pm 1.09}$	N/A	$16.51_{\pm 1.53}$
	texture	50	$71.27_{\pm 1.99}$	$71.17_{\pm 3.89}$	$57.41_{\pm 3.33}$	$62.78_{\pm 4.21}$	$65.24_{\pm 4.52}$	$69.45_{\pm 2.15}$	$62.93_{\pm 4.84}$	$64.33_{\pm 3.57}$	N/A	**75.79**$_{\pm 3.07}$
		100	$80.40_{\pm 2.45}$	$80.38_{\pm 2.67}$	$65.63_{\pm 4.21}$	$75.38_{\pm 3.99}$	$77.67_{\pm 2.62}$	$79.31_{\pm 1.88}$	$75.98_{\pm 2.56}$	$77.30_{\pm 2.44}$	N/A	**82.30**$_{\pm 2.21}$
		200	$84.00_{\pm 1.56}$	$85.12_{\pm 3.07}$	$76.98_{\pm 2.25}$	$84.44_{\pm 2.41}$	$85.30_{\pm 1.96}$	$84.00_{\pm 1.20}$	$83.70_{\pm 2.05}$	$80.02_{\pm 1.60}$	N/A	**85.92**$_{\pm 2.18}$
		500	$89.43_{\pm 0.80}$	$90.17_{\pm 1.25}$	$88.97_{\pm 1.44}$	$90.00_{\pm 1.66}$	$89.99_{\pm 1.06}$	$90.17_{\pm 1.32}$	**91.01**$_{\pm 1.32}$	$88.98_{\pm 1.26}$	N/A	$90.77_{\pm 1.10}$
Average rank			$4.36_{\pm 1.95}$	$3.02_{\pm 1.14}$	$6.88_{\pm 2.25}$	$7.82_{\pm 1.61}$	$8.45_{\pm 1.95}$	$6.12_{\pm 2.23}$	$7.85_{\pm 1.77}$	$6.00_{\pm 1.75}$	$3.12_{\pm 1.75}$	**1.38**$_{\pm 0.57}$

Table 12: **Classification accuracy** (%) of XGBoost, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean ± std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable or the downstream predictor failed to converge, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. TabEBM achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	19.70$_{\pm 6.33}$	N/A	19.44$_{\pm 4.11}$	17.32$_{\pm 2.75}$	18.11$_{\pm 3.07}$	16.15$_{\pm 3.80}$	20.71$_{\pm 5.24}$	17.40$_{\pm 4.89}$	24.00$_{\pm 3.64}$	**24.18**$_{\pm 3.05}$
		50	39.01$_{\pm 4.92}$	37.68$_{\pm 5.40}$	33.07$_{\pm 4.18}$	24.38$_{\pm 3.45}$	23.09$_{\pm 4.55}$	30.87$_{\pm 5.70}$	34.13$_{\pm 6.45}$	33.62$_{\pm 3.67}$	39.78$_{\pm 6.03}$	**44.46**$_{\pm 4.97}$
		100	57.59$_{\pm 3.69}$	60.16$_{\pm 5.75}$	49.23$_{\pm 5.51}$	43.33$_{\pm 7.92}$	37.69$_{\pm 5.96}$	48.36$_{\pm 4.08}$	43.97$_{\pm 5.45}$	47.00$_{\pm 3.29}$	53.74$_{\pm 7.94}$	**62.77**$_{\pm 5.85}$
		200	74.05$_{\pm 2.92}$	76.90$_{\pm 4.96}$	69.71$_{\pm 4.28}$	67.46$_{\pm 4.39}$	58.29$_{\pm 7.96}$	69.68$_{\pm 4.23}$	63.69$_{\pm 6.32}$	66.09$_{\pm 4.78}$	73.19$_{\pm 6.06}$	**79.25**$_{\pm 3.83}$
		500	88.89$_{\pm 1.71}$	90.02$_{\pm 1.51}$	90.10$_{\pm 1.80}$	89.37$_{\pm 1.81}$	86.03$_{\pm 2.31}$	87.29$_{\pm 2.08}$	90.05$_{\pm 2.70}$	85.04$_{\pm 2.07}$	89.66$_{\pm 1.17}$	**91.81**$_{\pm 1.44}$
	fourier	20	10.00$_{\pm 0.00}$	N/A	14.64$_{\pm 3.13}$	13.58$_{\pm 2.57}$	13.82$_{\pm 4.14}$	11.72$_{\pm 4.19}$	16.38$_{\pm 3.86}$	12.34$_{\pm 3.59}$	23.50$_{\pm 1.56}$	**26.78**$_{\pm 4.82}$
		50	42.10$_{\pm 6.19}$	43.40$_{\pm 5.22}$	34.32$_{\pm 3.98}$	24.68$_{\pm 6.47}$	17.66$_{\pm 4.64}$	24.82$_{\pm 6.35}$	27.74$_{\pm 5.86}$	35.42$_{\pm 7.51}$	35.60$_{\pm 3.11}$	**45.08**$_{\pm 6.47}$
		100	54.84$_{\pm 2.78}$	52.92$_{\pm 5.69}$	48.22$_{\pm 3.28}$	36.90$_{\pm 5.15}$	30.36$_{\pm 3.94}$	42.46$_{\pm 4.13}$	40.28$_{\pm 3.41}$	48.78$_{\pm 4.36}$	49.80$_{\pm 1.98}$	**54.94**$_{\pm 5.72}$
		200	63.88$_{\pm 3.35}$	65.34$_{\pm 3.57}$	58.36$_{\pm 3.27}$	53.20$_{\pm 5.26}$	46.96$_{\pm 4.58}$	61.40$_{\pm 4.12}$	52.10$_{\pm 3.32}$	56.66$_{\pm 2.67}$	66.60$_{\pm 4.24}$	**67.68**$_{\pm 3.19}$
		500	74.56$_{\pm 1.97}$	74.18$_{\pm 2.10}$	68.28$_{\pm 2.82}$	67.98$_{\pm 2.07}$	61.24$_{\pm 2.35}$	72.78$_{\pm 2.54}$	67.50$_{\pm 2.57}$	68.28$_{\pm 3.43}$	N/A	**76.25**$_{\pm 3.18}$
	biodeg	20	62.95$_{\pm 7.95}$	66.51$_{\pm 5.84}$	62.72$_{\pm 5.69}$	55.24$_{\pm 6.28}$	59.20$_{\pm 7.83}$	54.65$_{\pm 5.56}$	62.78$_{\pm 5.98}$	61.09$_{\pm 10.49}$	65.52$_{\pm 6.08}$	**66.64**$_{\pm 6.71}$
		50	67.96$_{\pm 3.45}$	67.69$_{\pm 4.42}$	66.22$_{\pm 5.70}$	61.64$_{\pm 6.73}$	60.72$_{\pm 5.73}$	57.48$_{\pm 8.28}$	**69.48**$_{\pm 5.35}$	65.93$_{\pm 4.98}$	67.76$_{\pm 4.90}$	67.90$_{\pm 3.27}$
		100	**73.88**$_{\pm 2.55}$	72.05$_{\pm 4.75}$	72.11$_{\pm 3.17}$	70.41$_{\pm 3.60}$	66.02$_{\pm 6.25}$	69.35$_{\pm 4.66}$	71.11$_{\pm 3.88}$	69.03$_{\pm 4.33}$	72.58$_{\pm 2.91}$	71.05$_{\pm 5.70}$
		200	76.38$_{\pm 4.85}$	74.98$_{\pm 3.15}$	73.93$_{\pm 3.29}$	75.68$_{\pm 4.15}$	67.82$_{\pm 3.91}$	72.58$_{\pm 5.07}$	74.74$_{\pm 2.24}$	73.84$_{\pm 3.82}$	75.85$_{\pm 1.80}$	**76.74**$_{\pm 2.44}$
		500	78.45$_{\pm 3.37}$	79.38$_{\pm 1.99}$	78.88$_{\pm 3.42}$	**80.15**$_{\pm 1.87}$	76.72$_{\pm 3.44}$	77.10$_{\pm 2.96}$	78.14$_{\pm 2.65}$	78.83$_{\pm 2.21}$	79.40$_{\pm 1.49}$	78.80$_{\pm 3.76}$
	steel	20	53.12$_{\pm 5.62}$	55.64$_{\pm 4.76}$	53.32$_{\pm 7.25}$	55.36$_{\pm 6.24}$	52.38$_{\pm 3.55}$	52.44$_{\pm 4.08}$	51.34$_{\pm 4.15}$	50.74$_{\pm 2.53}$	55.43$_{\pm 5.57}$	**55.78**$_{\pm 4.53}$
		50	66.73$_{\pm 9.11}$	60.79$_{\pm 5.52}$	59.51$_{\pm 4.15}$	54.82$_{\pm 4.23}$	54.79$_{\pm 4.69}$	59.71$_{\pm 6.94}$	57.66$_{\pm 5.19}$	55.89$_{\pm 4.50}$	63.78$_{\pm 7.20}$	**74.18**$_{\pm 13.67}$
		100	83.17$_{\pm 9.36}$	66.95$_{\pm 6.51}$	61.72$_{\pm 6.80}$	65.12$_{\pm 3.02}$	60.56$_{\pm 4.37}$	72.02$_{\pm 12.47}$	59.67$_{\pm 4.77}$	59.04$_{\pm 4.76}$	90.52$_{\pm 7.47}$	**96.55**$_{\pm 2.66}$
		200	95.94$_{\pm 2.73}$	81.21$_{\pm 5.01}$	73.14$_{\pm 5.45}$	70.64$_{\pm 10.67}$	70.26$_{\pm 9.25}$	74.50$_{\pm 23.57}$	74.57$_{\pm 9.36}$	65.41$_{\pm 6.70}$	99.14$_{\pm 1.19}$	**99.54**$_{\pm 0.62}$
		500	99.95$_{\pm 0.10}$	97.04$_{\pm 2.14}$	95.27$_{\pm 2.88}$	89.46$_{\pm 6.88}$	83.25$_{\pm 8.10}$	91.72$_{\pm 15.34}$	87.59$_{\pm 6.72}$	79.54$_{\pm 15.29}$	100.00$_{\pm 0.00}$	100.00$_{\pm 0.00}$
	stock	20	76.42$_{\pm 4.34}$	78.92$_{\pm 5.21}$	67.46$_{\pm 13.93}$	60.56$_{\pm 9.69}$	73.36$_{\pm 9.57}$	77.45$_{\pm 9.80}$	69.15$_{\pm 9.35}$	70.88$_{\pm 8.52}$	79.82$_{\pm 4.52}$	**83.44**$_{\pm 3.74}$
		50	83.71$_{\pm 3.40}$	86.23$_{\pm 2.54}$	84.65$_{\pm 4.44}$	79.31$_{\pm 6.58}$	76.27$_{\pm 3.89}$	85.70$_{\pm 3.96}$	81.61$_{\pm 1.97}$	84.98$_{\pm 4.44}$	87.28$_{\pm 3.65}$	**88.21**$_{\pm 3.31}$
		100	88.19$_{\pm 3.04}$	89.01$_{\pm 2.07}$	85.66$_{\pm 6.01}$	84.68$_{\pm 2.87}$	82.50$_{\pm 3.73}$	**90.07**$_{\pm 3.41}$	86.09$_{\pm 4.08}$	84.67$_{\pm 7.29}$	90.01$_{\pm 3.46}$	89.66$_{\pm 3.28}$
		200	**92.32**$_{\pm 1.35}$	92.26$_{\pm 2.33}$	90.94$_{\pm 1.98}$	89.01$_{\pm 2.53}$	88.92$_{\pm 2.67}$	91.36$_{\pm 3.79}$	91.04$_{\pm 1.46}$	91.42$_{\pm 2.66}$	91.72$_{\pm 2.77}$	92.17$_{\pm 1.51}$
More than 10 classes	energy	50	12.05$_{\pm 2.42}$	N/A	11.60$_{\pm 3.83}$	14.47$_{\pm 5.32}$	10.95$_{\pm 4.68}$	10.21$_{\pm 5.11}$	12.81$_{\pm 2.51}$	12.34$_{\pm 3.55}$	N/A	**21.07**$_{\pm 3.99}$
		100	**29.37**$_{\pm 1.72}$	N/A	20.61$_{\pm 5.39}$	19.81$_{\pm 4.52}$	22.71$_{\pm 6.15}$	22.27$_{\pm 2.12}$	22.02$_{\pm 3.54}$	10.01$_{\pm 3.40}$	N/A	27.93$_{\pm 4.16}$
		200	**44.96**$_{\pm 3.31}$	N/A	36.73$_{\pm 6.03}$	35.92$_{\pm 8.45}$	33.71$_{\pm 6.54}$	34.73$_{\pm 5.89}$	37.06$_{\pm 5.26}$	18.81$_{\pm 7.27}$	N/A	40.95$_{\pm 5.59}$
	collins	100	7.77$_{\pm 2.21}$	N/A	7.76$_{\pm 0.95}$	6.52$_{\pm 1.16}$	6.11$_{\pm 1.09}$	**8.95**$_{\pm 1.90}$	6.21$_{\pm 1.14}$	5.96$_{\pm 1.07}$	N/A	8.73$_{\pm 1.64}$
		200	10.58$_{\pm 2.57}$	11.46$_{\pm 2.11}$	9.43$_{\pm 2.20}$	9.84$_{\pm 1.56}$	8.26$_{\pm 1.75}$	9.80$_{\pm 1.96}$	8.90$_{\pm 0.83}$	9.79$_{\pm 0.80}$	N/A	**11.72**$_{\pm 1.34}$
	texture	50	56.72$_{\pm 6.12}$	60.99$_{\pm 4.35}$	45.76$_{\pm 6.50}$	39.50$_{\pm 6.46}$	43.02$_{\pm 6.12}$	50.22$_{\pm 6.28}$	43.71$_{\pm 5.98}$	46.21$_{\pm 7.95}$	N/A	**69.11**$_{\pm 3.27}$
		100	68.96$_{\pm 2.59}$	69.77$_{\pm 4.63}$	54.95$_{\pm 5.99}$	55.52$_{\pm 7.80}$	63.23$_{\pm 4.80}$	65.59$_{\pm 3.62}$	57.04$_{\pm 6.59}$	62.06$_{\pm 6.11}$	N/A	**76.35**$_{\pm 2.64}$
		200	77.91$_{\pm 1.98}$	81.55$_{\pm 2.22}$	70.70$_{\pm 4.40}$	71.60$_{\pm 4.19}$	73.76$_{\pm 5.69}$	77.06$_{\pm 2.17}$	72.56$_{\pm 4.09}$	70.31$_{\pm 6.55}$	N/A	**82.59**$_{\pm 2.15}$
		500	89.37$_{\pm 1.11}$	**89.87**$_{\pm 1.24}$	85.06$_{\pm 2.40}$	86.80$_{\pm 2.25}$	86.83$_{\pm 1.89}$	86.52$_{\pm 1.66}$	85.70$_{\pm 2.75}$	87.07$_{\pm 2.43}$	N/A	89.69$_{\pm 1.10}$
Average rank			3.64$_{\pm 2.09}$	3.45$_{\pm 1.48}$	6.32$_{\pm 1.86}$	7.33$_{\pm 2.19}$	8.64$_{\pm 1.82}$	6.30$_{\pm 2.44}$	6.64$_{\pm 2.18}$	7.62$_{\pm 1.84}$	3.44$_{\pm 1.54}$	**1.62**$_{\pm 1.29}$

Table 13: **Classification accuracy** (%) of TabPFN, comparing data augmentation on eight real-world tabular datasets with varied real data availability. We report the mean ± std balanced accuracy and average accuracy rank across datasets. A higher rank implies higher accuracy. Note that "N/A" denotes that a specific generator was not applicable or the downstream predictor failed to converge, and the rank is computed with the mean balanced accuracy of other methods. We **bold** the highest accuracy for each dataset of different sample size. TabEBM achieves the best overall performance against Baseline and benchmark generators.

	Datasets	N_{real}	Baseline (Real data)	SMOTE	TVAE	CTGAN	NFLOW	TabDDPM	ARF	GOGGLE	TabPFGen	**TabEBM**
At most 10 classes	protein	20	$27.80_{\pm 4.37}$	N/A	$19.21_{\pm 3.80}$	$20.58_{\pm 4.63}$	$20.80_{\pm 4.34}$	$18.89_{\pm 4.37}$	$23.97_{\pm 3.05}$	$10.55_{\pm 1.61}$	$33.42_{\pm 5.95}$	$\mathbf{34.63}_{\pm 5.78}$
		50	$55.24_{\pm 3.46}$	$\mathbf{59.85}_{\pm 3.87}$	$43.58_{\pm 6.20}$	$37.37_{\pm 8.01}$	$34.42_{\pm 6.65}$	$21.70_{\pm 7.43}$	$46.02_{\pm 2.60}$	$13.54_{\pm 4.22}$	$57.63_{\pm 2.82}$	$58.88_{\pm 3.99}$
		100	$74.31_{\pm 3.49}$	$\mathbf{80.05}_{\pm 3.16}$	$68.15_{\pm 5.54}$	$71.10_{\pm 2.55}$	$57.89_{\pm 6.13}$	$59.28_{\pm 8.23}$	$65.84_{\pm 3.03}$	$23.69_{\pm 10.40}$	$77.60_{\pm 4.03}$	$78.26_{\pm 3.75}$
		200	$88.67_{\pm 1.53}$	$\mathbf{91.79}_{\pm 1.42}$	$87.05_{\pm 2.85}$	$86.69_{\pm 2.85}$	$83.29_{\pm 2.42}$	$87.39_{\pm 2.95}$	$85.49_{\pm 2.49}$	$77.63_{\pm 6.11}$	$90.77_{\pm 1.37}$	$90.94_{\pm 1.46}$
		500	$97.31_{\pm 0.69}$	$\mathbf{97.69}_{\pm 0.77}$	$97.51_{\pm 0.85}$	$97.58_{\pm 0.85}$	$96.89_{\pm 0.62}$	$97.44_{\pm 0.85}$	$97.40_{\pm 0.60}$	$97.35_{\pm 0.61}$	$97.24_{\pm 0.80}$	$97.28_{\pm 0.62}$
	fourier	20	$30.06_{\pm 6.85}$	N/A	$22.00_{\pm 4.62}$	$20.10_{\pm 4.31}$	$14.52_{\pm 3.96}$	$12.22_{\pm 2.40}$	$21.64_{\pm 5.91}$	$14.64_{\pm 3.94}$	N/A	$36.56_{\pm 4.96}$
		50	$53.62_{\pm 4.71}$	$53.08_{\pm 3.34}$	$45.82_{\pm 4.29}$	$37.46_{\pm 5.82}$	$28.78_{\pm 2.78}$	$22.74_{\pm 5.11}$	$42.14_{\pm 3.09}$	$11.30_{\pm 1.50}$	$53.15_{\pm 3.50}$	$\mathbf{53.82}_{\pm 3.92}$
		100	$64.62_{\pm 4.14}$	$63.66_{\pm 3.92}$	$56.68_{\pm 3.02}$	$54.78_{\pm 2.80}$	$45.50_{\pm 4.50}$	$49.36_{\pm 8.51}$	$54.74_{\pm 2.78}$	$21.40_{\pm 4.29}$	$\mathbf{65.95}_{\pm 3.49}$	$65.40_{\pm 3.61}$
		200	$71.62_{\pm 2.59}$	$70.56_{\pm 3.61}$	$66.48_{\pm 3.82}$	$66.14_{\pm 4.02}$	$62.64_{\pm 2.60}$	$72.12_{\pm 2.64}$	$65.04_{\pm 3.20}$	$52.18_{\pm 7.35}$	$69.93_{\pm 3.91}$	$\mathbf{72.48}_{\pm 3.08}$
		500	$77.66_{\pm 1.61}$	$77.50_{\pm 1.08}$	$76.80_{\pm 1.34}$	$77.82_{\pm 1.24}$	$73.90_{\pm 1.76}$	$\mathbf{79.16}_{\pm 2.05}$	$75.70_{\pm 2.11}$	$74.36_{\pm 2.53}$	$77.30_{\pm 0.42}$	$77.40_{\pm 1.28}$
	biodeg	20	$65.26_{\pm 8.01}$	$68.72_{\pm 4.50}$	$69.02_{\pm 5.37}$	$59.39_{\pm 6.25}$	$58.28_{\pm 8.30}$	$50.00_{\pm 0.00}$	$58.45_{\pm 8.16}$	$51.80_{\pm 4.07}$	$70.68_{\pm 4.94}$	$\mathbf{71.18}_{\pm 5.25}$
		50	$75.27_{\pm 2.63}$	$74.65_{\pm 3.28}$	$73.44_{\pm 4.02}$	$70.21_{\pm 3.61}$	$55.68_{\pm 9.27}$	$50.00_{\pm 0.00}$	$72.74_{\pm 3.74}$	$55.75_{\pm 7.45}$	$\mathbf{75.69}_{\pm 2.44}$	$75.56_{\pm 3.22}$
		100	$78.92_{\pm 1.98}$	$77.78_{\pm 2.65}$	$77.27_{\pm 3.15}$	$77.71_{\pm 1.81}$	$63.50_{\pm 10.77}$	$57.50_{\pm 6.27}$	$77.25_{\pm 1.66}$	$65.87_{\pm 6.72}$	$78.15_{\pm 1.45}$	$\mathbf{79.00}_{\pm 1.99}$
		200	$\mathbf{82.59}_{\pm 1.84}$	$81.42_{\pm 1.27}$	$80.48_{\pm 1.82}$	$80.19_{\pm 2.48}$	$79.16_{\pm 2.49}$	$80.45_{\pm 1.48}$	$80.88_{\pm 1.68}$	$80.66_{\pm 1.49}$	$82.56_{\pm 1.68}$	$82.58_{\pm 1.90}$
		500	$\mathbf{85.00}_{\pm 0.70}$	$84.37_{\pm 0.75}$	$84.40_{\pm 0.68}$	$84.67_{\pm 0.98}$	$84.45_{\pm 0.91}$	$84.58_{\pm 0.70}$	$84.68_{\pm 1.06}$	$83.66_{\pm 0.67}$	$84.56_{\pm 0.98}$	$84.55_{\pm 0.92}$
	steel	20	$56.77_{\pm 4.17}$	$55.95_{\pm 4.30}$	$56.03_{\pm 4.37}$	$55.62_{\pm 4.80}$	$52.52_{\pm 4.64}$	$50.00_{\pm 0.00}$	$52.39_{\pm 3.13}$	$50.05_{\pm 0.17}$	$64.80_{\pm 5.66}$	$\mathbf{65.87}_{\pm 6.14}$
		50	$82.34_{\pm 8.38}$	$63.42_{\pm 3.93}$	$62.08_{\pm 2.69}$	$63.98_{\pm 4.08}$	$52.92_{\pm 4.72}$	$50.64_{\pm 2.01}$	$61.32_{\pm 4.55}$	$50.36_{\pm 1.09}$	$84.70_{\pm 7.84}$	$\mathbf{86.30}_{\pm 6.73}$
		100	$97.37_{\pm 1.37}$	$73.06_{\pm 4.46}$	$71.96_{\pm 5.40}$	$72.23_{\pm 4.15}$	$56.34_{\pm 6.30}$	$80.87_{\pm 20.44}$	$69.29_{\pm 5.70}$	$51.18_{\pm 3.24}$	$97.49_{\pm 1.21}$	$\mathbf{97.81}_{\pm 1.49}$
		200	$98.84_{\pm 0.70}$	$82.32_{\pm 2.88}$	$81.78_{\pm 3.36}$	$83.24_{\pm 2.68}$	$82.92_{\pm 6.21}$	$\mathbf{99.35}_{\pm 0.70}$	$86.40_{\pm 4.22}$	$64.42_{\pm 11.35}$	$98.80_{\pm 0.73}$	$98.96_{\pm 0.71}$
		500	$99.74_{\pm 0.29}$	$94.27_{\pm 2.39}$	$94.93_{\pm 1.89}$	$96.98_{\pm 1.34}$	$98.32_{\pm 1.19}$	$\mathbf{99.88}_{\pm 0.15}$	$95.70_{\pm 1.50}$	$98.56_{\pm 0.52}$	$99.77_{\pm 0.30}$	$99.74_{\pm 0.29}$
	stock	20	$83.18_{\pm 4.37}$	$83.69_{\pm 3.10}$	$74.01_{\pm 5.09}$	$56.92_{\pm 16.52}$	$74.99_{\pm 6.60}$	$78.73_{\pm 12.25}$	$69.64_{\pm 6.88}$	$73.40_{\pm 4.88}$	$82.95_{\pm 4.44}$	$\mathbf{83.81}_{\pm 4.94}$
		50	$90.01_{\pm 2.07}$	$90.01_{\pm 2.43}$	$82.27_{\pm 4.30}$	$78.91_{\pm 4.14}$	$78.94_{\pm 8.78}$	$89.68_{\pm 1.92}$	$83.72_{\pm 2.50}$	$79.00_{\pm 6.87}$	$89.95_{\pm 2.08}$	$\mathbf{90.15}_{\pm 1.76}$
		100	$92.39_{\pm 1.06}$	$92.09_{\pm 1.45}$	$90.75_{\pm 2.20}$	$89.43_{\pm 3.29}$	$86.16_{\pm 3.83}$	$92.12_{\pm 1.16}$	$90.17_{\pm 1.92}$	$89.30_{\pm 1.33}$	$92.12_{\pm 1.12}$	$\mathbf{92.57}_{\pm 1.27}$
		200	$94.16_{\pm 0.92}$	$93.99_{\pm 0.70}$	$93.57_{\pm 1.10}$	$93.28_{\pm 1.59}$	$91.92_{\pm 2.00}$	$\mathbf{94.22}_{\pm 1.10}$	$93.05_{\pm 1.35}$	$92.07_{\pm 1.76}$	$94.17_{\pm 0.89}$	$94.16_{\pm 1.07}$
Average rank			$3.08_{\pm 1.22}$	$4.23_{\pm 2.32}$	$6.12_{\pm 1.57}$	$6.29_{\pm 2.07}$	$8.42_{\pm 1.32}$	$6.12_{\pm 3.38}$	$6.54_{\pm 1.61}$	$8.83_{\pm 1.46}$	$3.12_{\pm 1.80}$	$\mathbf{2.23}_{\pm 1.83}$

E.4 Results on computation efficiency

Figure 9 shows the trade-off between accuracy and the time needed for generating stratified synthetic data (for data augmentation). We measure the total duration of (i) training the model and (ii) generating 500 synthetic samples. The results show that TabEBM is practical, as it achieves higher downstream accuracy with lower time costs.

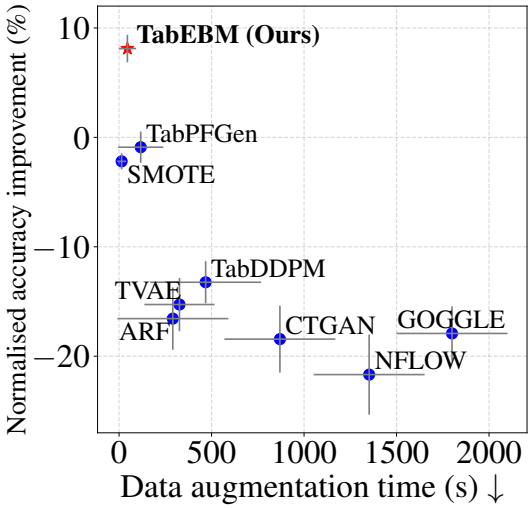


Figure 9: **Median data augmentation time vs. mean normalised balanced accuracy.** TabEBM achieves higher downstream accuracy with lower computation costs. TabEBM typically operates 3-30 times faster than most benchmark generative models.